Copy No. 15

DEFENCE R&D DÉFENSE

DSSP20 (BETA EDITION) USER GUIDE TO THE PREPROCESSING MODULES

M. Mackay

DEFENCE RESEARCH ESTABLISHMENT ATLANTIC

Technical Memorandum

DREA TM 1999-108

August 1999

National Defence Défense nationale

Canada

101-03405/

National Defence
Research and Development Branch

Défense nationale
Bureau de recherche et développement

DREA TM 1999-108

DSSP-20 (Beta Edition) Users Guide To the Processing Modules

M.Mackay

DEFENCE RESEARCH ESTABLISHMENT ATLANTIC
PO Box 1012, Dartmouth
Nova Scotia, Canada
B2Y 3Z7

Approved by J.L. Kennedy
Head, Warship Signature and Safety

August 1999

TECHNICAL MEMORANDUM

Prepared by

Defence Research Establishment Atlantic

Centre de Recherches pour la Défense Atlantique

Canada

Table of Contents

			Page No.
		Abstract	ii
		Executive Summary	iii
		Table of Contents	iv
		Nomenclature	vi
1		Introduction	1
2		General Considerations	2
	2.1	File Nomenclature	2
	2.2	User Input	3
	2.3	Axis Systems	4
	2.4	Applicability	4
3		DSSP20A: Basic Vehicle Definition	5
	3.1	Input and Output Files	5
	3.2	Component Geometry and Miscellaneous Parameters	5
	3.3	Input File Specification	9
	3.4	Level 1 Keywords	10
	3.5	Level 2 Hull Component Keywords	11
	3.5.1	Data Block Keywords	11
	3.5.2	Hull Axis Keywords	12
	3.5.3	Other Parameter Keywords	15
	3.5.4	Example	15
	3.6	Level 2 Lifting Appendage Keywords	16
	3.6.1	Data Block Keyword	17
	3.6.2	Appendage Type Keywords	17
	3.6.3	Control Type Keywords	19
	3.6.4	Endplate Specification Keywords	19
	3.6.5	Save Keywords	20
	3.6.6	Copy Keywords	20
	3.6.7	Other Parameter Keywords	20
	3.6.8	Examples	21
	3.7	Plot Format	22
	3.8	Reference Axes	25
	3.9	Interactive Dialogue	26
	3.10	Output	26
4		DSSP20I: Inertial and Added Mass Estimates	27
	4.1	Input and Output Files	27
	4.2	Interactive Dialogue	28
	4.3	Output	28
5		DSSP20V: Vehicle System Definition	29
	5.1	Input and Output Files	29
	5.2	Input File Specification	29

Page No.

5.3 Level 1 Keywords 30
5.4 Level 2 Lifting Appendage Control Data 30
5.5 Level 2 Propulsor Data 31
5.5.1 Propulsor Type: Thrust and Torque Estimation 32
5.5.2 Geometric Arrangement 32
5.5.3 Handedness 33
5.5.4 Other Parameters 33
5.6 Additional Propulsor Data at Level 1 34
5.7 Interactive Dialogue 34
5.8 Output 35
6 Concluding Remarks 36
Appendix A: Parser Dictionary List 37
Appendix B: *Root*.AI and *Root*.VI Input Summaries 38
Appendix C: Error Messages 44
References 58

Nomenclature

A_B	hull base area
A_t	hull station area
b	semispan of isolated appendage, or as measured from the hull centerline
b_{ep}	spanwise location of endplate
b_{exp}	exposed appendage span
$b_{sp(exp)}$	spanwise location of sailplanes on exposed sail
b_1, b_2	spanwise location of inboard and outboard flap extent
B_t	hull station breadth
$c, \overline{c}$	appendage chord, mean chord
$\overline{cf}$	mean flap chord
$ci,\ i = 1, 4$	appendage planform corner points
CG	vehicle center of gravity
C_X, C_Z	axial and normal force coefficients
C_{Xj}	junction axial force coefficient
C_{X0}	appendage zero lift axial force coefficient
$C_{X\alpha\alpha}$	appendage induced drag axial force coefficient
$C_{Z\alpha}$	appendage linear normal force coefficient
$C_{Z\alpha\alpha\alpha}$	appendage nonlinear (crossflow drag) normal force coefficient
$C_{\Delta f}$	hull residual drag coefficient
D_M	maximum hull diameter
D_P	propeller diameter
f	appendage camber
F_t	hull station camber
h	total endplate height
h'	endplate height from attachment
J_S	advance coefficient based on vehicle speed
k_r	hull roughness allowance
K_T	propulsor thrust coefficient
K_Q	propulsor torque coefficient
k_δ	appendage flap profile drag
$k_{\eta t}$	tail efficiency
L	vehicle length
n	propulsor RPM
n_M	maximum RPM
$\dot{n}_M$	RPM rate limit
n_{segH}	number of hull cross section segments for plotting
n_{segL}	number of appendage cross section segments for plotting
n_0	self-propulsion RPM
P_P	propeller pitch
r	local hull radius
r_M	maximum hull radius
R_c	chord Reynolds number

S_{exp}	exposed appendage area
S_m	area of moving part of all-moving appendage
t	appendage thickness
T_i	hull station height
u	axial component of speed
U_o	ahead speed for propulsor initialization
W_{ID}, W_{IH}, W_{IR}	weights for indirect depth, heading, and roll control
x, y, z	vehicle body axes
x'	distance downstream from origin of hull boundary layer
$x_{ci}, y_{ci}, z_{ci}, i = 1, 4$	appendage corner point coordinates
x_m, y_m, z_m	hull midpoint coordinates
x_n, y_n, z_n	hull nose coordinates
x_p, y_p, z_p	propulsor location coordinates
x_t, y_t, z_t	hull tail coordinates
x_0, y_0, z_0	vehicle reference origin
X, Z	axial and normal forces
α	geometric angle of attack
α_b	appendage geometric twist
$\alpha_{(1)}, \alpha_{(2)}, \alpha_{(3)}$	equivalent angles of attack for local appendage forces
δ	geometric angle of deflection
δ_e	effective flap angle of deflection
δ_M	maximum control deflection angle
$\dot{\delta}_M$	control deflection rate limit
ζ_n	RPM rate damping
ζ_δ	control deflection damping
η	appendage flap gap correction
θ_P	propulsor local axis pitch
Θ	axis rotation in pitch
ρ	water density
ϕ	appendage anhedral or dihedral angle
ϕ_R	angle of rotation for a copied appendage
ϕ_0, θ_0, ψ_0	vehicle reference axis directions
Φ	axis rotation in roll
ψ_P	propulsor local axis yaw
Ψ	axis rotation in yaw
ω_n	RPM rate frequency
ω_δ	control response frequency

1 INTRODUCTION

DREA Submarine Simulation Program, Version 2.0 (DSSP20), is a suite of software modules for evaluating underwater vehicle hydrodynamics and manoeuvering. The key calculations of hydrodynamic forces and moments on the vehicle are done by component based semiempirical methods which are sufficiently accurate for preliminary design and routine manoeuvering analysis [1]. Development is continuing to further improve these methods and widen their scope of application. The code is currently in beta edition; a number of features are not fully implemented or are partially complete, including time history effects, flooding and ballast blowing models, and autopilot control. Nevertheless, the beta edition — specifically, Edition 981009 — is documented here because it is a useful tool in its own right and is a stable benchmark for future releases.

A flowchart of DSSP20 is sketched in figure 1; the flow of logic is predominantly from left to right. Preprocessing functions of DSSP20 consist of vehicle input checking and calculation of as many global quantities and coefficients as possible. The tool kit comprises subroutines that calculate hydrodynamic forces and moments, and track the time history of wake trajectories and other quantities. There is no direct user interaction with the tool kit; client software sets up modeling and simulation problems and solves them by calling appropriate subroutines from tool kit and utility libraries. As the name implies, client software is conceptually an external entity and not truly part of DSSP20. However, modules for single vehicle captive and free simulations have been implemented for the program, and are documented in a separate report [2].

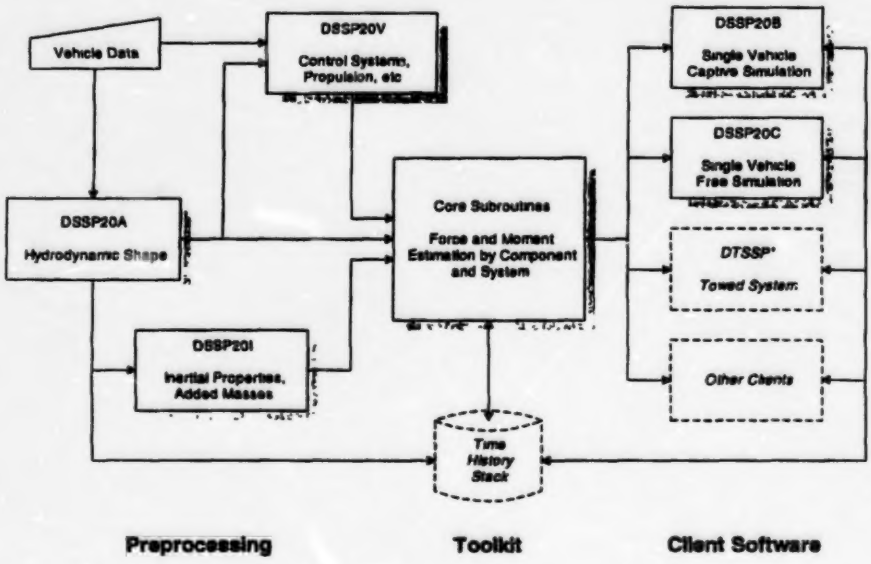

Fig 1 Flowchart of DSSP20. Items in dashed lines are projected or incomplete.

Each module of DSSP20 runs as a stand-alone program. For portability, the program user interface is minimal; most modules require a brief interactive dialogue using the keyboard and monitor. The code was written in FORTRAN 77 (with a couple of minor extensions) using the MS PowerStation Version 4.0 compiler for the PC. It has also been compiled and validated on a SGI Unix workstation.

As shown in figure 1, preprocessing is done by DSSP20A, DSSP20I, and DSSP20V:

DSSP20A takes in the geometric definition of the vehicle together with some parameters that characterize hydrodynamic interference effects, and produces the basic vehicle description. It must be run before any other module.

DSSP20I estimates the inertial and added mass properties of the vehicle from the output of DSSP20A. It is required in all cases except for purely static simulations.

DSSP20V processes vehicle system information in a way analogous to DSSP20A with the geometric information. It is currently used to specify the propulsor(s) and control surface parameters, and will be used in subsequent releases for the autopilot and other systems. It is required when any of these systems or functions have to be modeled.

Preprocessing is required only once for each vehicle unless some aspect of its configuration or definition is changed. The resulting internal description of the vehicle, comprising data files that are accessed directly by the tool kit, is then available for any client software module.

The main objective of this user guide is to facilitate preparation of input files for the preprocessing modules. Minimum input requirements are quite simple, but optional parameters are available to accommodate a wide range of vehicle configurations. New users are advised to first read this guide completely, with particular attention to section 2, General Considerations, and to the numerous examples throughout the text. As a reference, more experienced users will need only sections 3, 4, and 5 (guides to modules DSSP20A, DSSP20I, and DSSP20V respectively), and the appendices, which contain additional information including input summaries and lists of error messages and their interpretation.

2 GENERAL CONSIDERATIONS

2.1 File Nomenclature

DSSP20 uses a standardized naming scheme for its input and output files. The filename is *Root.Ext* in which *Root* is a user-defined string up to eight characters long ("DSSP20" is used by default). For module DSSP20?, where '?' is 'A', 'I', or 'V' in the present context, file extensions *Ext* are interpreted as follows:

File	Function
Root.?I	DSSP20? user-prepared input file
Root.?G	DSSP20? graphics (plotting) output file
Root.?L	DSSP20? run log output file

Root.?O DSSP20? output file to other DSSP20 modules
Root.?P DSSP20? printer output file

All output files are in ASCII characters and can be read (and manipulated) with any text editor. However, *Root*.?O files, when read, are checked for consistency with the corresponding *Root*.AO file, and a change made in them will result in a fatal error. If modification of the output is required, the module must be run again with appropriately modified input.

2.2 User Input

The input character set for DSSP20 comprises upper- and lower-case letters, the space, digits 0 to 9, and the "printable" characters `*()-=+;:',<.>/'~!@#$%^&_\|[{]}"`. Tabs or other unrecognized characters in the input generally result in a fatal error.

In order to allow the user-prepared input files (*Root*.?I) to be more readable, a simple input parsing scheme is employed. Each line in the input file is treated as a separate record of up to 71 characters consisting of words separated by one or more delimiters and terminated by an optional end-of-record character. Input is case-insensitive. The delimiters are the space ' ', comma ',', and semicolon ';'. The end-of-record (EOR) characters are the double quote '"', exclamation point '!', and percentage sign '%'. A word is a string of any other characters. Input following an EOR character is ignored and may therefore constitute a comment. Any input that is encountered after the input requirements of a record are fulfilled is similarly ignored.

Input is controlled by keywords; a keyword consists of an alphabetic string of up to ten characters contained in the parser's dictionary. A typical input record consists of a keyword followed by one or more numerical arguments. Since arguments may be optional, it is prudent to always preceed a following comment by an EOR character.

Keywords may be abbreviated by truncation on input so long as the result is not ambiguous, i.e., the abbreviation may not be the abbreviation of another word in the dictionary. For example, referring to the dictionary list in appendix A we see that 'Len' is an acceptable abbreviation for 'Length', whereas 'Le' is not since the intended keyword could be 'Left'. On the other hand, an unabbreviated keyword takes priority over any abbreviated alternatives — 'Delta', 'P', and others are not considered to be ambiguous but are interpreted at face value. For this reason, keyword abbreviation should be used with care since the result may be unintended and inappropriate.

With keyword-driven input, records can in principle appear in any order in the input file. However, in DSSP20 this freedom is limited to within an input 'level'. Input is initially at Level 1 and is switched to Level 2 and above by specific keywords. Another context where input order is important is within a data block, i.e., a number of physical records that DSSP20 treats as a single logical record. Both these concepts are best illustrated by the DSSP20A input description given in sections 3.4 to 3.8.

Numerical values are also parsed. An integer input can have no more than 20 characters. It is read, right justifed, in FORTRAN I20 format. A floating-point input can have no more than 30 characters. Initially it is read, right justified, in I30 format and converted to a double-precision variable; if that fails, it is re-read in F30.0 format into a double precision variable.

The following are examples of valid input records defining the value of density, ρ :

```
Rho 1025.     ! Density in kg/m**3
RHO 1025      " Rho has only one argument, so ...
rho 1.025e3     no EOR character is required before the comment.
```

Blank lines are ignored, and purely comment lines should start with an EOR character otherwise the parser will try to interpret the first delimited string as a keyword. Appendix B gives input file summaries that illustrate some of the concepts discussed in this section.

2.3 Axis Systems

DSSP20 uses SI units; coordinates and lengths are in metres. There are two principal axis systems: vehicle axes, in which vehicle geometry and most other input is defined, and reference axes, in which output quantities such as hydrodynamic forces and moments are defined Both systems are righthanded. Vehicle axes are determined indirectly by the (x, y, z) coordinates of vehicle geometry which are input in module DSSP20A. All components of the vehicle must use vehicle axes for their definition. By default, reference axes are the same as vehicle axes, but they may be redefined by the user To be most meaningful, the reference axes should coincide with conventional hydrodynamic axes [3], i e., with the origin in the vicinity of the CG and with x directed forward, y to starboard, and z downward.

Internally, the program uses a separate local axis system for each component. For a hull component the origin is at its longitudinal CB, local x is directed forward along the longitudinal axis, and y and z are determined from vehicle axes or are user-defined. For a lifting appendage the origin is at the root mid-chord, local y is outward along the mid-chord line, z is normal to the effective plane of the appendage, and x is obtained from their cross product. In some instances, local axes are used for printed output. The principal reason for the user to be aware of them is that the deflection of flaps and all-moving appendages is defined in local axes: positive for rotation of z into x.

Examples of the various axis systems are sketched in figure 2. In this case, the vehicle axis origin is at the bow of the submarine. Local axes for the sail and sternplanes are not shown to avoid complicating the figure. For all-moving rudders on the same shaft, the definition of appendage local axes results in deflections of opposite sign on the upper and lower rudders; the same observation applies to port and starboard sternplanes.

2.4 Applicability

It is implicit in many algorithms used in the current version of DSSP20 that the vehicle is streamined and at moderate incidence, i.e., that vehicle components are reasonably slender and have their hydrodynamic axis — the longitudinal axis of a hull or the chordline of a lifting appendage — oriented more-or-less in the direction of motion (or against the direction of incident flow). The program therefore cannot model bluff or irregular underwater vehicles such as ROVs, but can

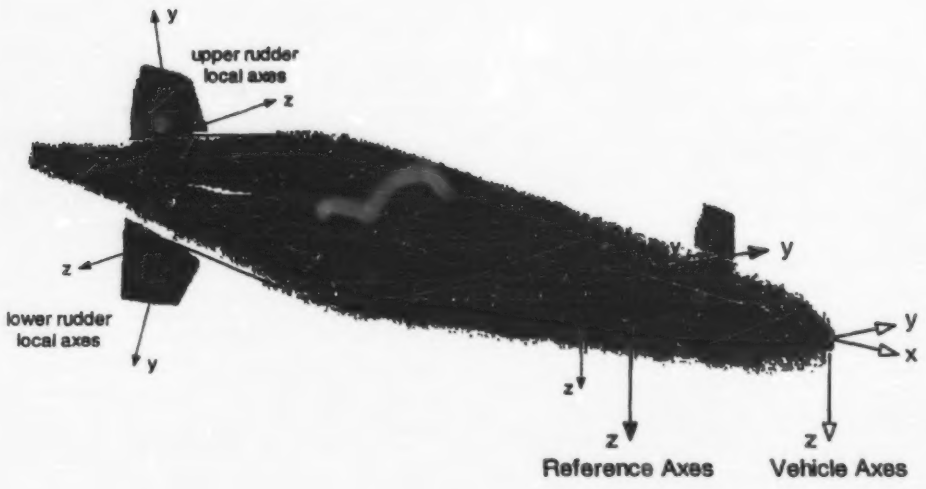

Fig 2 Example axis systems.

accomodate submarines, AUVs, and towfish. The geometry is currently described by up to two hull components and up to twelve lifting appendages; there can be up to three propulsors. Future versions will add component types for more diverse applications such as the semi-submersible Dolphin snort mast [4].

3 DSSP20A: BASIC VEHICLE DEFINITION

3.1 Input and Output Files

DSSP20A input and output files follow the nomenclature introduced in section 2.1:

File	Function
Root.AI	DSSP20A user-prepared input file
Root.AG	DSSP20A graphics (plotting) output file
Root.AL	DSSP20A run log output file
Root.AO	DSSP20A output file to other DSSP20 modules
Root.AP	DSSP20A printer output file

3.2 Component Geometry and Miscellaneous Parameters

The purpose of this section is to provide an overview of the geometric and other details required to set up a vehicle description with DSSP20A. Basic input defines the component envelope in terms of a number of section properties, e.g., hull breadth and height. Since vehicle components have no

knowledge of their relationship to each other, interference calculations need additional parameters such as the hull radius at the attachment point of an appendage. A few non-geometric parameters are needed to complete the definition.

The nomenclature used to define basic hull component geometry is illustrated in figure 3 (the component images on this and the following figure are from screen-shots of *Root*.`AG` output in AcroSpin format, see section 3.7). A hull is defined at 21 stations equally spaced from nose to tail The minimum input required is diameter at each station for an axisymmetric hull, and direct or indirect input of the nose and tail points in vehicle axes. The tail section may be closed, with zero diameter, or truncated, with a nonzero diameter. In the latter case, base drag can be modelled. Alternative input for each station may include breadth and height (for non-axisymmetry), cross-section area, and camber (e.g., to represent asymmetry due to a deck casing). Values of residual drag coefficient ($C_{\Delta f}$) or roughness allowance (k_r) may be input to override `DSSP20A` default values.

The nomenclature used to define basic lifting appendage geometry is illustrated in figure 4. The four corner points $c1, c2, c3$, and $c4$ must be given in vehicle axes. Appendage sections are assumed to be NACA 4–digit, and thickness/chord ratio and other parameters are optional. An isolated appendage is assumed to be mounted on a reflecting plane at the root, i.e., loading is finite at the root because of lift carry-over. Therefore, an unattached symmetrical wing would be modelled by two isolated appendages, one for each semispan. All other appendage configurations require a type specification (`Rudder`, `Sail`, etc.) in order to model interference effects.

Figures 5, 6, and 7 illustrate a number of the additional geometric parameters needed to account for interference; they are defined in the nomenclature. According to appendage type, the local angle

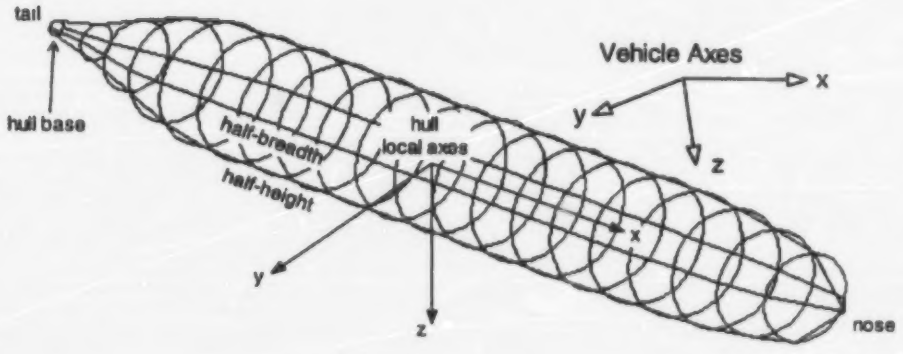

Fig 3 Hull definition nomenclature.

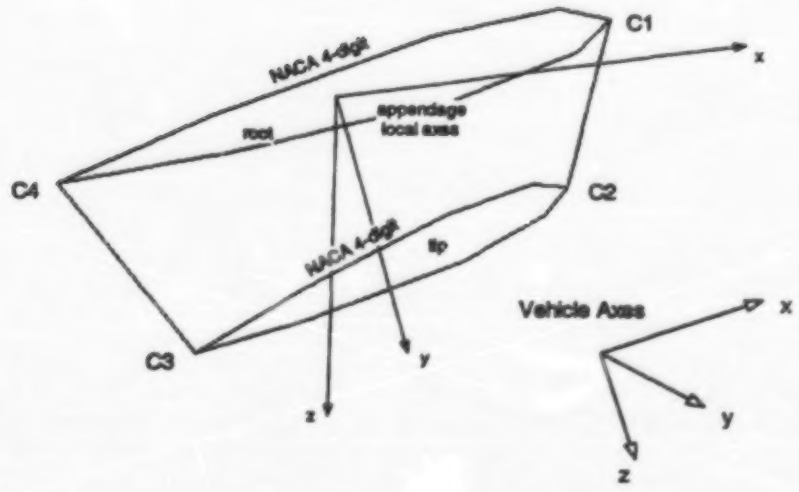

Fig 4 Appendage definition nomenclature.

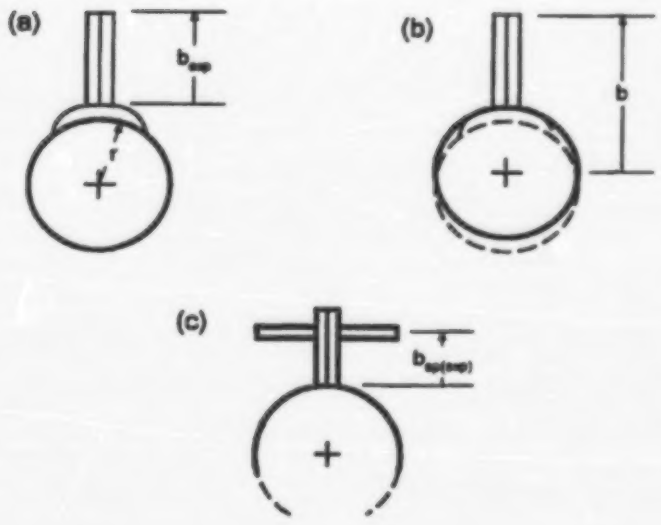

Fig 5 Hull/sail input nomenclature: (a) non-axisymmetric hull, (b) equivalent axisymmetric hull, and (c) sailplane location.

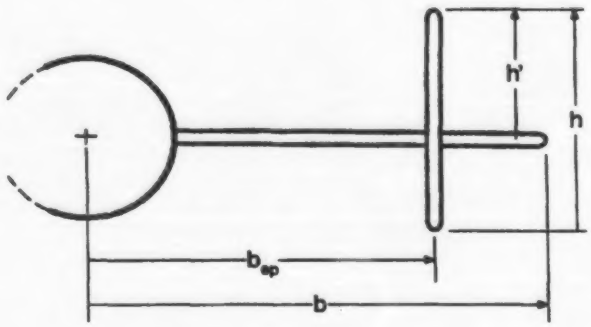

Fig 6 Endplate input nomenclature.

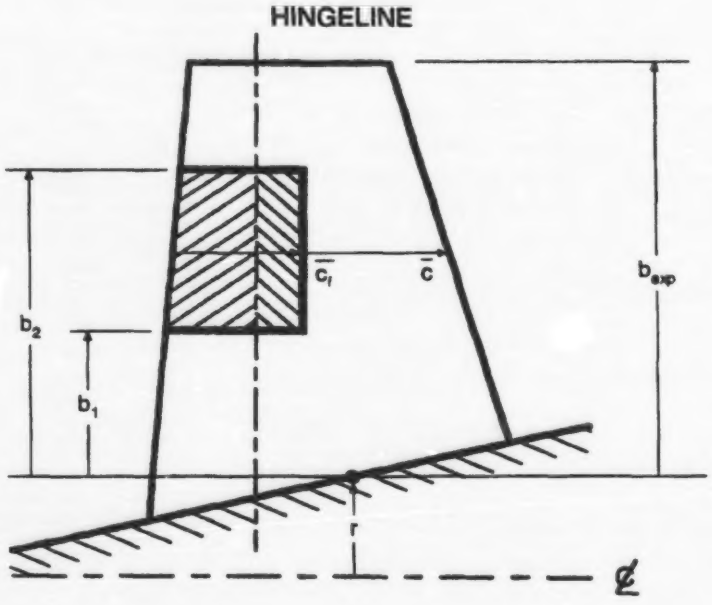

Fig 7 Flapped appendage input nomenclature.

of attack is modified so that the appendage normal (Z) and axial (X) forces, in nondimensional coefficients, are:

$$C_Z = C_{Z\alpha}\,\alpha_{(1)} + C_{Z\alpha\alpha\alpha}\,\alpha_{(2)}^3$$
$$C_X = C_{X0} + C_{X\alpha\alpha}\,\alpha_{(3)}^2 + \ldots$$

where $\alpha_{(1)}$, $\alpha_{(2)}$, and $\alpha_{(3)}$ are equivalent angles of attack which are functions of local geometric angles of attack and deflection, α, δ, and of the interference parameters:

$$\alpha_{(1)} = \alpha_{(1)}(\alpha, \delta, r, b_{exp}, b_1, b_2, \ldots)$$
etc.

The estimation of interference for a submarine sail is described in reference [5].

A number of non-geometric parameters may be defined where it is necessary to override default values assigned by DSSP20. These include coefficients for junction drag (C_{Xj}) and zero lift drag (C_{X0}), a flap gap correction (η), flap profile drag (k_δ), and tail efficiency ($k_{\eta t}$). The full description of lifting appendage input is given in section 3.6.

3.3 Input File Specification

In *Root*.AI, Level 1 input is used to flag the vehicle components — descriptions of which are then input at Level 2 — and to provide overall program control. Return to Level 1 from Level 2 occurs when an out-of-context keyword is encountered.

Before describing the input records in detail, it is useful to consider a short example of *Root*.AI that illustrates a number of the features of user input previously discussed. We have noted that there are few restrictions on format, but systematic indentation and spacing improves readability as shown. The italicised comments to the right are not part of the file.

```
Title DARPA SUBOFF model with hull and sail   Level 1

Plot                                          Level 1 → Level 2
  Maple                                       Level 2 ...
                                              Blank lines are ignored.
Reference                 * (See Roddy)       The EOR is required since Reference
 -2.01296  0.  0.                                   has an optional argument.
  0.       0.  0.

Hull
  Label #Hull
  Nose  0.0
  Tail -4.26085
  Diameter                                    This starts a data block comprising
    0.          0.359225    0.430175             three floating-point values per record
    0.477678    0.504277    0.508                on the next seven records.
```

```
    0.508        0.508        0.508
    0.508        0.508        0.508
    0.508        0.508        0.508
    0.508        0.496741     0.418993
    0.283524     0.139928     0.05969      End of data block.

Lift             Comment: this is a sail  An EOR is not required in this case
  Label #Sail
  Sail   0.254
  Planform                                 Another data block; no
   -0.9244541  0. -0.254      " (approx)      intervening blank records are
   -0.9244541  0. -0.4595812                 permitted before the end.
   -1.2927541  0. -0.4595812
   -1.2927541  0. -0.254                    End of data block.
  ToC 0.1810345                            Last record in file.
```

Root.AI input is summarised in appendix B.

3.4 Level 1 Keywords

The following table summarizes the keyword records at Level 1 of DSSP20A input.

Keyword	Argument(s)	Function
Hull		announces hull component input (Level 2)
Lift		announces lifting component input (Level 2)
Plot		generate plot file (optional Level 2 input)
Reference	[*keyword*]	define reference axes (by keyword or Level 2 input)
Title	*Title String*	title for printer output; up to 64 characters

These records are all optional and may appear in any order on the file, although required Level 2 input must immediately follow a keyword record.

There are two keywords for component input: Hull and Lift. Neither supports an argument, although all 71 characters, including comments, of the record are echoed to *Root*.AP for indentification purposes. Thus

```
Lift - starboard bowplane
```

is reproduced verbatim in the printer output. Encountering a component input keyword at Level 1 causes program control to switch to the appropriate subroutine which reads and processes the corresponding Level 2 input for that component.

Hull

Announces Level 2 hull input as described in Section 3.5.

Lift

Announces Level 2 lifting appendage input as described in Section 3.6.

The other Level 1 keywords are:

`Plot`

Announces plot file format information as described in Section 3.7; without this record a plot file is not generated.

`Reference` [*keyword*]

Permits user override of the default reference axes and origin as described in section 3.8.

`Title` *Title String*

A title string of up to 64 characters is preceeded by the keyword `Title`. The string is not parsed, and may therefore contain any character including delimiters and end-of-record characters. However, the normal end-of-record ("Enter" or "Carriage Return") terminates the string.

3.5 Level 2 Hull Component Keywords

The following table summarizes the keyword records used to characterize hull components. The arguments are all real values; optional arguments are enclosed in brackets, [ ].

Keyword	Argument(s)	Function
`BaseArea`	A_B	define hull base area
`Diameter`		diameter data block follows
`Label`	*Label String*	component label
`Length`	L	define component length
`LoD`	L/D_M	define length/diameter ratio
`MidHull`	x_m , [y_m , z_m]	define midpoint
`Nose`	x_n , [y_n , z_n]	define nose point
`Pitch`	Θ	define rotation in pitch
`ResiDragR`	$C_{\Delta f}$ or *keyword*	define hull residual drag coefficient or method
`Roll`	Φ	define rotation in roll
`RoughAllow`	k_r	define hull roughness allowance
`Station`		station data block follows
`Tail`	x_t , [y_t , z_t]	define tail point
`Yaw`	Ψ	define rotation in yaw

A keyword record other than one of these will terminate input for the current component: processing of Level 2 data is completed, the input stream is backspaced, and the new record is re-read at Level 1.

Hull component keyword records are discussed below in terms of the functional group (i.e., data block, component axis, and so on) that they belong to.

3.5.1 Data Block Keywords

The keywords which announce a data block are `Diameter` and `Station`. There is no argument to the keyword, but the records immediately following, which constitute the data block, must contain the required data — blank records are not permitted within the data block or between the keyword record and the data block.

`Diameter`

The next seven records contain values of diameter for an axisymmetric hull, three values per record for 21 equally spaced stations from nose to tail. The data are read in as breadth, B:

$B_1 \quad B_2 \quad B_3$

...

$B_i \quad B_{i+1} \quad B_{i+2}$

...

$B_{19} \quad B_{20} \quad B_{21}$

In generating the hull geometry, $T_i = B_i$ and $A_i = \pi B_i^2/4$, where T is height and A is cross-section area.

`Station`

The next 21 records contain values of hull breadth B, T (optional), A (optional), and z-wise camber F (also optional) for 21 equally spaced stations from nose to tail. Hull camber, by analogy with appendage section camber, is station mid-height offset from the nose–tail line, positive downwards. It is primarily applicable to older irregular submarine hulls; e.g., the example in 3.5.4 includes camber. If F is omitted, $F_i = 0$. If there are only two values per record, $A_i = \pi B_i T_i/4$; if there is only one value on a record, it represents diameter for an axisymmetric hull ($T_i = B_i$ and $A_i = \pi B_i^2/4$). The section data block format is therefore:

$B_1 \quad [\, T_1 \quad [\, A_1 \quad [\, F_1 \,]\,]\,]$

...

$B_i \quad [\, T_i \quad [\, A_i \quad [\, F_i \,]\,]\,]$

...

$B_{21} \quad [\, T_{21} \quad [\, A_{21} \quad [\, F_{21} \,]\,]\,]$

where arguments are enclosed in nested brackets, [ ], to indicate that they are progressively optional from right to left.

3.5.2 Hull Axis Keywords

The hull component location and attitude, and therefore the definition of its local axis system, is determined by a combination of the keywords `Length`, `MidHull`, `Nose`, `Pitch`, `Roll`, `Tail`, and `Yaw`. The local x axis is directed forward and passes through the hull tailpoint, midpoint, and nosepoint. However, the hull may have camber on its z axis (see 3.5.1), in which case `MidHull` is simply the mean of `Nose` and `Tail`. To fully define hull location and attitude, the user must provide an unambiguous combination of component axis records. Redundant data are checked for consistency — inconsistency generates a fatal error. By default, the axis rotations (taken in order Ψ, Θ, Φ) are assumed to be zero. Some examples are given at the end of this section.

`Length` L

Defines component length between tailpoint and nosepoint. `Length` plus one of `Nose`, `MidHull`, or `Tail` provides an unambiguous hull axis definition for the default (zero) rotation. Rotation angles may be defined as required to override the default values.

`MidHull` x_m , [y_m , z_m]

Defines the midpoint of the hull component in vehicle coordinates; y_m and z_m are optional, and default to zero. `MidHull` plus one of `Nose` or `Tail` provides an unambiguous hull definition including yaw and pitch rotation angles; however, a `Roll` record is explicitly required to override the roll angle default. `MidHull` plus `Length` assumes the default (zero) rotation angles unless one or more of `Yaw`, `Pitch`, or `Roll` is also provided.

`Nose` x_n , [y_n , z_n]

Defines the nose point of the hull component in vehicle coordinates; y_n and z_n are optional, and default to zero. `Nose` plus one of `MidHull` or `Tail` provides an unambiguous hull definition including yaw and pitch rotation angles; however, a `Roll` record is explicitly required to override the roll angle default. `Nose` plus `Length` assumes the default (zero) rotation angles unless one or more of `Yaw`, `Pitch`, or `Roll` is also provided.

`Pitch` Θ

Defines pitch rotation of the local axis system with respect to the vehicle axis system, in degrees. Rotations are taken in the conventional order: yaw, pitch, roll. A value for Θ will be calculated if at least two records with the keyword `Nose`, `MidHull`, or `Tail` are present. Otherwise Θ will take the default value (zero) if a `Pitch` record is not provided.

`Roll` Φ

Defines roll rotation of the local axis system with respect to the vehicle axis system, in degrees. Rotations are taken in the conventional order: yaw, pitch, roll. If a `Roll` record is not provided, Φ will take the default value (zero).

`Tail` x_t , [y_t , z_t]

Defines the tail point of the hull component in vehicle coordinates; y_t and z_t are optional, and default to zero. `Tail` plus one of `Nose` or `MidHull` provides an unambiguous hull definition including yaw and pitch rotation angles; however, a `Roll` record is explicitly required to override the roll angle default. `Tail` plus `Length` assumes the default (zero) rotation angles unless one or more of `Yaw`, `Pitch`, or `Roll` is also provided.

`Yaw` Ψ

Defines yaw rotation of the local axis system with respect to the vehicle axis system, in degrees. Rotations are taken in the conventional order: yaw, pitch, roll. A value for Ψ will be calculated if at least two records with the keyword `Nose`, `MidHull`, or `Tail` are present. Otherwise, Ψ will take the default value (zero) if a `Yaw` record is not provided. Note that Ψ cannot be calculated if the input hull points are such that pitch is ± 90 degrees (the indeterminate $\tan^{-1}(0/0)$ is generated); in this case Ψ should be input explicitly, as in the last two examples below.

The following examples illustrate combinations of these keywords:

i. A hull component of length 10.829485 with midpoint located at $(-2.1461, -0.23341, 0.7705)$ in the vehicle axes. By default, local axes are aligned with vehicle axes.

```
MidHull  -2.1461  -.23341  .7705
Length   10.829485
```

ii. The same component given explicit rotations so that local axes are no longer aligned with the vehicle axes:

```
MidHull  -2.1461  -.23341   .7705
Length   10.829485
Yaw      -2.484513
Pitch    17.958312
Roll     27.5
```

iii The previous component with two hull points defined, and therefore having implicit rotations in yaw and pitch. This and the previous example are exactly equivalent within truncation error:

```
Nose      3       -.4567   -.899
Tail     -7.2922  -.01012  2.44
Roll     27.5
```

iv. Again, the same configuration, but with redundant information in pitch and yaw:

```
Nose      3       -.4567   -.899
Tail     -7.2922  -.01012  2.44
Yaw      -2.484513
Pitch    17.958312
Roll     27.5
```

Redundancy is overlooked; however, a small change or error in the data, such entering z_n as $-.889$ instead of $-.899$ in the previous example, will generate a fatal error because of inconsistency in input.

v. A hull component of length 5.3025 with midpoint located at (0, 0, 0), rotated implicitly by 90 degrees in pitch. This generates a fatal error as noted under the `Yaw` keyword:

```
Nose    0   0   -2.65125
Tail    0   0    2.65125
```

vi. The error in the previous example is avoided by explicitly defining the pitch angle:

```
MidHull  0  0  0
Length   5.3025
Pitch    90
```

3.5.3 Other Parameter Keywords

Parameter keywords serve a number of purposes including to override default values and internal calculations. Two of these, `ResiDragR` and `RoughAllow`, are related to hydrodynamic calulations rather than geometric setup, but they are included here because they derive from physical properties of the hull.

`BaseArea` A_B
: This record redefines the hull base area, which defaults to A_{21}, for the hull base drag calculation. It is required that $0 \leq A_B \leq \max A_i$.

`Label` *Label String*
: This record provides a user-defined component label. The label string consists of a hash or question symbol ('#' or '?') followed by up to 24 alphabetic characters. DSSP20 ignores case in label strings. If there is no `Label` record, the default label is `#HULL`*nnn* where *nnn* is the number of this hull component as encountered in the input file (i.e., *nnn* = 001 for the first, 002 for the second, etc.).

`LoD` L/D_M
: This record overrides the calculated value of length/diameter ratio for the current hull component. The argument is mandatory, and has a permitted range of $4 \leq L/D_M \leq 16$.

`ResiDragR` $C_{\Delta f}$ or *keyword*
: This record defines a hull residual drag coefficient $C_{\Delta f}$ or the method for its estimation. The two methods available are the semiempirical expression given by Hoerner [6] for bodies of rotation (*keyword* = `Hoerner`), or a fit to data on torpedo-like bodies (*keyword* = `Torpedo`). If the numerical argument is given, then it is required that $0 \leq C_{\Delta f} \leq 0.5$. The default for hull residual drag is calculation by Hoerner's method.

`RoughAllow` k_r
: This record defines a coefficient k_r for the hull roughness allowance in hull drag. By default, $k_r = 0.0004$; the user-input value must be $0 \leq k_r \leq 0.001$.

3.5.4 Example

The following example of hull component input illustrates full station input:

```
Hull   non-axisymmetric irregularly cambered hull
Label  #OldHull
  Length 90
  Nose   0
  Station
       0         0         0        -4.5124
       3.1852    5.8493    14.6329  -1.5878
       4.2905    6.4693    21.7999  -1.2778
       5.863     7.7136    35.5195  -0.65561
       6.8402    7.765     41.7158  -0.62988
       7.3708    7.7568    44.9042  -0.6184
       7.7754    7.7369    47.2476  -0.60513
       7.9554    7.7099    48.1727  -0.57359
```

```
      8.06      7.674    48.5788    -0.54651
      8.087     7.629    48.4557    -0.52401
      8.0247    7.5824   47.7887    -0.51575
      7.9026    7.5385   46.7891    -0.50927
      7.6091    7.5092   44.8763    -0.49464
      7.2071    7.4757   42.3158    -0.4779
      6.7391    7.4397   39.3774    -0.45989
      5.9536    7.3471   34.3546    -0.42646
      5.1436    7.2502   29.2892    -0.39185
      4.2332    5.2498   17.4543    -1.2784
      3.2382    4.6716   11.8812    -1.422
      2.2389    4.5178    7.9442    -1.2875
      0         0         0         -1.8607
RoughAllow  0.0008
```

3.6 Level 2 Lifting Appendage Keywords

The following table summarizes the keyword records used to characterize lifting appendages. The arguments are all real values; optional arguments are enclosed in brackets, [ ].

Keyword	Argument(s)	Function
`AllMoving`	$[S_m/S_{exp}]$	define control type as All-Moving
`Camber`	f	apply geometric camber
`CXJ`	C_{X_J}	define junction axial force coefficient
`CXZero`	C_{X0}	define zero lift axial force coefficient
`DelNonLin`		enable nonlinear flap calculation
`FlapGap`	η	define flap balance and gap correction
`Flapped`	$[b_1/b_{exp}\ ,\ b_2/b_{exp}]\ ,\ [\overline{cf}/\overline{c}]$	define control type as Flapped
`FlaProDrag`	k_δ	define flap profile drag
`HasEndP`	$h/b\ ,\ [b_{ep}/b]$	correct for having an endplate on this appendage
`IsEndP`	$[h'/h]$	correct for this appendage being an endplate
`Kill`		save for a copy, but do not use, this appendage
`Label`	*Label String*	component label
`LBowplane`	r	define appendage type as Large Bowplane
`Planform`		appendage corner data block follows
`Reflected`		copy saved appendage by reflecting
`Rotated`	ϕ_R	copy saved appendage by rotating
`Rudder`	$r\ ,\ [r_M]$	define appendage type as Rudder
`Sail`	r	define appendage type as Sail
`Sailplane`	$r\ ,\ b_{sp(exp)}/b_{exp}$	define appendage type as Sailplane
`Save`		save this appendage for subsequent copy
`SBowplane`	$r\ ,\ [\phi]$	define appendage type as Small Bowplane
`ShipStab`	$r\ ,\ [x']$	define appendage type as Ship Stabilizer

`Sternplane`	r , $[r_M]$	define appendage type as Sternplane
`Tail`	r , $[r_M]$	define appendage type as Tail
`TailEff`	$k_{\eta t}$	define tail efficiency
`ToC`	t/c	define section thickness/chord ratio
`Twist`	α_b	apply geometric twist

A keyword record other than one of these will terminate input for the current component: processing of Level 2 data is completed, the input stream is backspaced, and the new record is re-read at Level 1.

Keyword records are discussed below in terms of the functional group (i.e., data block, appendage type, and so on) that they belong to.

3.6.1 Data Block Keyword

At Level 2 appendage input, there is one keyword which announces a data block: `Planform`. There is no argument to the keyword, but the records immediately following, which constitute the data block, must contain the required data — blank records are not permitted within the data block or between the keyword record and the data block.

`Planform`

The next four records contain coordinates of the four appendage corner points in vehicle axes. The corner points are input one point per record, and must be in order: root leading edge, tip leading edge, tip trailing edge, and root trailing edge. The section data block format is therefore:

x_{c1} y_{c1} z_{c1}
x_{c2} y_{c2} z_{c2}
x_{c3} y_{c3} z_{c3}
x_{c4} y_{c4} z_{c4}

3.6.2 Appendage Type Keywords

The keywords which define appendage type are `LBowplane`, `Rudder`, `Sail`, `Sailplane`, `SBowplane`, `ShipStab`, `Sternplane`, and `Tail`. Defining the appendage type ensures that appropriate interference corrections are made to the appendage forces, and may have additional implications. For example, although interference corrections are identical for types `Rudder`, `Sternplane`, and `Tail`, these types respond differently to control commands in `DSSP20B` and `DSSP20C` [2]. Only one appendage type record may be supplied for each lifting component; if no type is specified, it is treated as an isolated appendage.

Appendage type records have at least one argument: a parameter r that is used to estimate lift carry-over. For the general case of an appendage mounted on a hull, and where local dihedral is minimal (i.e., the appendage is more-or-less normal to the hull surface), r is the hull radius at the appendage mid root chord.

The keyword `TailEff` is also discussed with this group because it pertains to only one class of appendage (`Rudder`/`Sternplane`/`Tail`).

`LBowplane` r

The appendage is defined as a large bowplane. Argument r is the mid-root hull radius; $r \geq 0$.

`Rudder` r, $[r_M]$

The appendage is defined as a rudder. Argument r is the mid-root hull radius; $r \geq 0$. Optional argument r_M is maximum hull radius (required for Dempsey's calculation of sternplane effectiveness [7]); by default $r_M = r$. Tail efficiency may be specified with a `TailEff` record.

`Sail` r

The appendage is defined as a sail. Argument r is the mid-root hull radius; $r \geq 0$.

`Sailplane` r, $b_{sp(exp)}/b_{exp}$

The appendage is defined as a sailplane; both arguments are mandatory. Argument r is the mid-root to sail centerline distance; $r \geq 0$. Parameters $b_{sp(exp)}$ and b_{exp} are exposed sailplane height (i.e, from the root) and exposed sail span, see figure 5. The permitted range of the argument is $0.0 \leq b_{sp(exp)}/b_{exp} \leq 1.0$.

`SBowplane` r, $[\phi]$

The appendage is defined as a small bowplane. Argument r is the mid-root hull radius; $r \geq 0$. Optional argument ϕ is the dihedral or anhedral angle with respect to the hull tangent, in degrees. The default value of ϕ is 0°, and its permitted range is $-90° \leq \phi \leq 90°$.

`ShipStab` r, $[x']$

The appendage is defined as a ship stabilizer. Argument r is the local hull radius; $r \geq 0$. Optional argument x' is downstream distance between the origin of the hull boundary layer (typically, the forwardmost point of the hull) and the origin of the appendage axis system, in metres, and must not be negative; x' defaults to zero.

`Sternplane` r, $[r_M]$

The appendage is defined as a sternplane. Argument r is the mid-root hull radius; $r \geq 0$. Optional argument r_M is maximum hull radius (required for Dempsey's calculation of sternplane effectiveness [7]); by default $r_M = r$. Tail efficiency may be specified with a `TailEff` record.

`Tail` r, $[r_M]$

The appendage is defined to be a tail fin. Argument r is the mid-root hull radius; $r \geq 0$. Optional argument r_M is maximum hull radius (required for Dempsey's calculation of sternplane effectiveness [7]); by default $r_M = r$. Tail efficiency may be specified with a `TailEff` record.

`TailEff` $k_{\eta t}$

This record gives a fixed value to override the estimation of tail efficiency, $k_{\eta t}$. By default, $k_{\eta t}$ is dependent on chord Reynolds number R_c. If $R_c \leq 1 \times 10^6$, $k_{\eta t}$ is calculated by Dempsey's method [7]; if $R_c \leq 2 \times 10^6$, it is set to 1.0; and it is varied linearly for R_c in between. This record is valid only for an appendage of type `Rudder`, `Sternplane`, or `Tail` The argument is mandatory and has a permitted range of $0.0 \leq k_{\eta t} \leq 1.5$.

3.6.3 Control Type Keywords

The keywords which define control type are `AllMoving` and `Flapped`. Keywords `DelNonLin`, `FlapGap`, and `FlaProDrag` are also discussed with this group because they are specific to a control surface with a trailing edge flap.

`AllMoving` $[S_m/S_{exp}]$

The appendage is an all-moving control surface; the argument is optional. Parameters S_m and S_{exp} are area of the moving part and total exposed area of the appendage. The default value of S_m/S_{exp} is 1.0, and its permitted range is $0.1 < S_m/S_{exp} \leq 1.0$.

`DelNonLin`

This record enables the calculation of effective flap angle, $\delta_e < \delta$, for an appendage with a trailing edge flap. The keyword record `Flapped` must also appear in the appendage specification. Without the `DelNonLin` record, $\delta_e = \delta$.

`FlapGap` η

This record overrides the default value of 1.0 for the flap balance and gap efficiency correction, η. It is valid only for an appendage with a trailing edge flap — the keyword record `Flapped` must also appear in the appendage specification. The argument is mandatory and has a permitted range of $0.8 \leq \eta \leq 1.25$.

`Flapped` $[b_1/b_{exp}\ ,\ b_2/b_{exp}]\ ,\ [\overline{c_f}/\overline{c}]$

The appendage is a control surface with a trailing edge flap. The arguments are the inner and outer spanwise flap fractions, and the chordwise flap fraction, see figure 7. (Note that $\overline{c}$ is sometimes defined from mid trailing edge to mid leading edge, and $\overline{c_f}$ is aligned accordingly; however, it is generally preferable to make them normal to the flap hinge line, as shown in the figure.) The first two arguments of `Flapped` are optional as a pair; the third argument is also optional. Thus, if only one argument is present, it is assumed to be $\overline{c_f}/\overline{c}$; if two are present, they are assumed to be b_1/b_{exp} and b_2/b_{exp}. Default values for the arguments are 0.0, 1.0, and 0.25 respectively, and their permitted ranges are $0.0 \leq b_1/b_{exp} \leq 1.0$, $b_1/b_{exp} \leq b_2/b_{exp} \leq 1.0$, and $0.0 \leq \overline{c_f}/\overline{c} \leq 1.0$.

`FlaProDrag` k_δ

This record overrides the default value of 0.0 for the coefficient of profile axial force (i.e., negative drag) associated with a trailing-edge flap deflection, k_δ (which is nondimensionalized with respect to exposed appendage area). It is valid only for an appendage with a trailing edge flap — the keyword record `Flapped` must also appear in the appendage specification. The argument is mandatory and has a permitted range of $-2.0 \leq k_\delta \leq 0.0$.

3.6.4 Endplate Specification Keywords

The keywords which control endplate specifications are `HasEndP` and `IsEndP`.

`HasEndP` $h/b\ ,\ [b_{ep}/b]$

The current appendage has an endplate; the first argument is mandatory, the second is optional. Parameters h, b, and b_{ep} are endplate height, appendage semispan, and spanwise location of the endplate (the last two measured from the hull centerline if appropriate), see

figure 6. Permitted range of the first argument is $0.0 \leq h/b \leq 2.5$. The default value of the second argument is 1.0 and its permitted range is $0.0 \leq b_{ep}/b \leq 1.0$.

`IsEndP` $[h'/h]$

The current appendage is an endplate; the argument is optional. Parameters h' and h are point of attachment relative to the full span and the full span ($h = 2b$), or height. The default value of the argument is 0.5, and its permitted range is $0.0 \leq h'/h \leq 1.0$.

3.6.5 Save Keywords

An appendage configuration can be saved, with or without itself being used, for use through reflection or rotation. The keywords which control saving are `Kill` and `Save`. Note that the component label is not part of the configuration; it is not saved, while all other attributes are.

`Kill`

The current appendage configuration will be saved for a subsequent copy, but will not itself be used. This will save the user the effort of calculating corner points for a non-vertical or non-horizontal appendage which is more easily defined in the vehicle xy or xz plane and then rotated into its correct orientation — see the first example in section 3.6.8.

`Save`

The current appendage configuration will be used and then saved for a subsequent copy.

3.6.6 Copy Keywords

The keywords which control copying a previously saved appendage are `Reflected` and `Rotated`. Only corner points are modified by a copy; all other physical attributes of the appendage remain the same. The label is not transferred, so that a `Label` record is required for the new appendage or the default (see 3.6.7) will be used.

These keywords cannot be used in combination with any others in Level 2 input except for a `Label` record.

`Reflected`

The saved appendage is reflected in the vehicle xz plane. This operation is illustrated by the second example in section 3.6.8.

`Rotated` ϕ_R

The saved appendage is rotated ϕ_R degrees about the vehicle x axis. The argument is mandatory; ϕ_R is positive rotating from the y axis to the z axis; there is no restriction on its value. This operation is illustrated by the first example in section 3.6.8.

3.6.7 Other Parameter Keywords

The keywords which fall into this category are `Camber`, `CXJ`, `CXZero`, `Label`, `ToC`, and `Twist`. They are in general applicable to any appendage type and in most cases serve to override default values of the lifting surface parameters.

`Camber` f

This record overrides the default value of 0.0 for nondimensional geometric section camber, f. The argument is mandatory, and is positive in the direction of increasing z_L in the local appendage axis system.

`CXJ` C_{Xj}

This record overrides the default value of 0.0 for the junction axial force coefficient, C_{Xj}, i.e., "junction drag". The argument is mandatory and must not be positive.

`CXZero` C_{X0}

This record overrides the default value of -0.01 for the zero lift axial force coefficient, C_{X0} (which is nondimensionalized with respect to appendage area). The argument is mandatory and must not be positive.

`Label` *Label String*

This record provides a user-defined component label. The label string consists of a hash or question symbol ('#' or '?') followed by up to 24 alphabetic characters. DSSP20 ignores case in label strings. If there is no `Label` record, the default label is `#LIFT`*nnn* where *nnn* is the number of this lifting appendage as encountered in the input file (i.e., *nnn* = 001 for the first, 002 for the second, etc.).

`ToC` t/c

This record overrides the default value of 0.12 for section thickness/chord ratio, t/c. The argument is mandatory and has a permitted range of $0.0 \leq t/c \leq 0.25$.

`Twist` α_t

This record overrides the default value of 0.0 for geometric section twist, α_t (angle of attack at the tip relative to the root). The argument is mandatory; its value is in degrees and has a permitted range of $-30° \leq \alpha_t \leq 30°$. Twist is positive in the sense of rotating the appendage local z axis into the x axis. This record is ignored, with a warning, for triangular planforms and for appendages which include twist in the input corner points.

3.6.8 Examples

A typical example of lift component input for X-rudders, in which a rudder is laid out in the vertical plane and then killed and rotated into the X-rudder positions, follows:

```
Lift  ! define tailplane in lower rudder position
      ! and kill for X-rudders copied by rotation
  Kill
  Tail   1.2998
  Planform
    -23.4000 0. 1.6899
    -24.4787 0. 4.65365
    -26.3250 0. 4.65365
    -26.3250 0. 0.9097
  AllMoving
  TailEff     0.66
  ToC         0.14
```

```
Lift  X-rudder - Port Lower
   Label #PortLower
   Rotate  45
Lift  X-rudder - Port Upper
   Label #PortUpper
   Rotate 135
Lift  X-rudder - Stbd Upper
   Label #StbdUpper
   Rotate 225
Lift  X-rudder - Stbd Lower
   Label #StbdLower
   Rotate 315
```

In the second example, a starboard sailplane is saved and its port counterpart is generated by reflection:

```
Lift  Stbd sailplane
   Label #StbdSailplane
   Save
   Sailplane  0.448165  1.0
   Planform
      14.625     0.0       -6.39844
      13.42731   3.290625  -6.39844
      12.675     3.290625  -6.39844
      12.675     0.896329  -6.39844
   AllMoving
   ToC  0.14

Lift  Port sailplane
   Label #PortSailplane
   Reflect
```

3.7 Plot Format

If a `Plot` record is found at Level 1 input, plotting output of the vehicle geometry is saved in ASCII format on file *Root*`.AG`. This enables the user to check that the geometry has been input correctly. Component intersections should be first checked with a wire frame plot using no hidden lines so that the details are not obscured.

Following the `Plot` record, the configuration of *Root*`.AG` is controlled by a number of optional input records at Level 2. Output file format is defined by a record containing one of the following keywords:

`AcroSpin`

> *Root*`.AG` is formatted for AcroSpin Version 2.0 for DOS. This is an old program but the file format is compatible with the "original DOS version of Acrospin" (actually version 2.3) that is marketed as a viewer for MathWare Derive (`http://www.mathware.com`). AcroSpin produces a wire-frame screen plot with fast zoom, rotation, and translation capabilities. An example screen-shot is shown in figure 8.

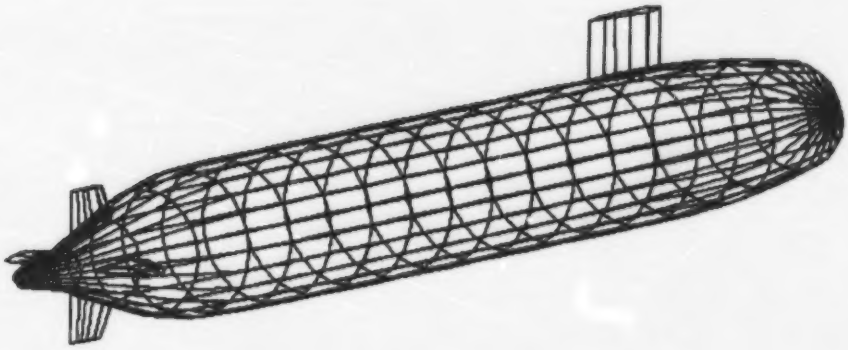

Fig 8 AcroSpin plot screen-shot.

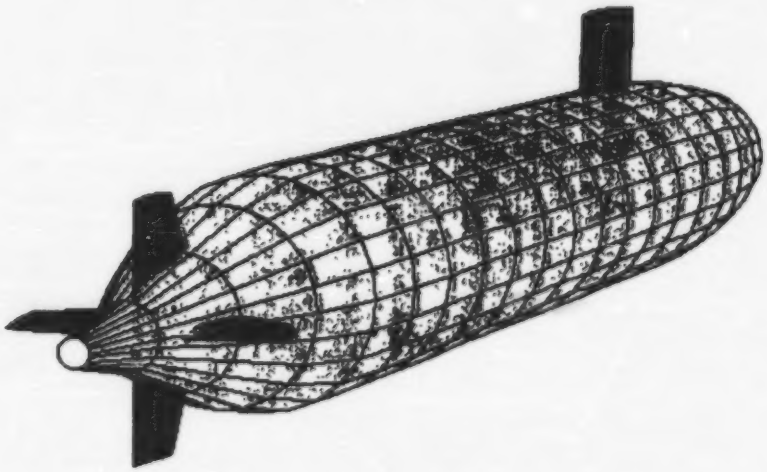

Fig 9 Maple plot screenshot; an axis box (here omitted) is optional.

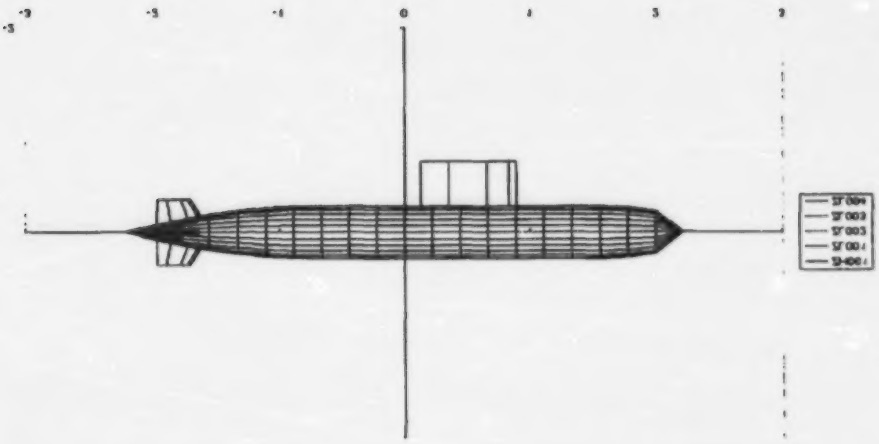

Fig 10 Spreadsht plot output plotted with MS Excel (flipped about the x axis to make the submarine upright) using default chart options.

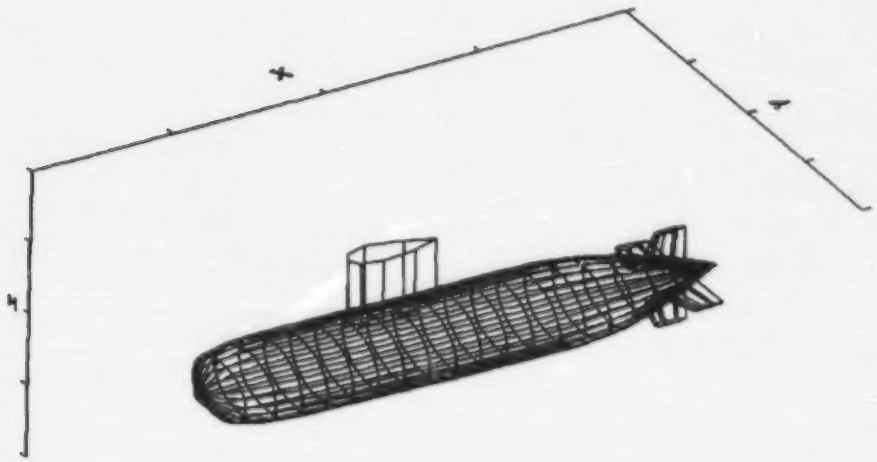

Fig 11 Spreadsht plot output plotted with Axum (flipped about the x axis to make the submarine upright) using default axes.

`Maple`

Root.`AG` is formatted for the DSSPplot Maple script written by George Watt at DREA. Called from a Waterloo Maple (`http://www.maplesoft.com`) worksheet, DSSPplot initializes a session for reading and plotting *Root*.`AG`. Options include solid modelling (patch plot) or wire frame with zoom, rotation, translation, and perspective capabilities. An example screen-shot is shown in figure 9.

`Spreadsht`

Root.`AG` is output in a generic format for plotting from a spreadsheet. It must be imported as an ASCII file, and for some applications must be renamed with a `.PRN` or `.TXT` extension. The vehicle components are tabulated in the spreadsheet as columns of (x,y,z) coordinates. The columns are labelled `XH001`, `YH001`, `ZH001`, `XL001`, and so on, where `H` denotes a hull, `L` denotes a lifting component, and the number is incremented for each additional component of the same type. Spreadsheets have the advantage of being widely available, but generally have limited plotting capability for the present application. Typical difficulties are a restriction to two dimensional line plotting, limited viewpoint control, and an inability to invert axis ranges (resulting in a "upside down" image). Subsequent image manipulation may alleviate the latter problems. In general, only wire frames can be plotted for this sort of data. An example screen-shot of a flipped (x,z) plot from Microsoft Excel is shown in figure 10. Somewhat more flexibility is found in spreadsheet-based plotting programs such as Axum (`http://www.mathsoft.com`). A screen-shot of a flipped three dimensional line plot from Axum is shown in figure 11.

The default *Root*.`AG` file format is currently `AcroSpin`; a VRML 1.0 output format is being developed as the future default.

Plotting requires component cross-sections, which are not defined or used elsewhere in `DSSP20`. Hull cross sections (21 equally spaced) are represented by an ellipse fitted to B_i and T_i; the number of perimeter segments is defined by parameter n_{segH}. By default, $n_{segH} = 24$. Lifting component cross sections (at tip and root only) are represented by a NACA 4-digit section fitted to their chord and maximum thickness; the number of perimeter segments is defined by parameter n_{segL}. By default, $n_{segL} = 8$. The value of these parameters may be redefined:

`HullSegmt` n_{segH}

This record overrides the default value of n_{segH}, the number of hull cross section segments. The argument is mandatory and must be one of 4, 6, 8, 12, or 24.

`LiftSegmt` n_{segL}

This record overrides the default value of n_{segL}, the number of lifting component cross section segments. The argument is mandatory and must be one of 2, 4, or 8.

A keyword record other than one of these will terminate input for the plotting file specification: processing of Level 2 data is completed, the input stream is backspaced, and the new record is re-read at Level 1.

3.8 Reference Axes

The following Level 1 record is required to override the `DSSP20A` default reference axes and origin:

`Reference` [*keyword*]

If this record has the keyword `CB` as an argument then there is no Level 2 input and the reference orgin is defined by the calculated CB of the vehicle and the reference axes are vehicle axes; otherwise, the reference origin and axes (angles of rotation from vehicle axes, in order, in degrees) must follow on the next two records as two triplets of real numbers:

$x_0 \quad y_0 \quad z_0$
$\psi_0 \quad \theta_0 \quad \phi_0$

These data records constitute a data block; that is, a blank record is not permitted between them, or between them and the `Reference` record.

3.9 Interactive Dialogue

At the start of execution, the user is asked to supply a character string *Root* for naming input and output files (see section 3.1):

```
Enter file name root (default = DSSP20):
```

A string up to eight characters long is expected; it should comply with the filename requirements of the host operating system. A carriage return indicates the default "DSSP20". If the input file cannot be found, the following message is issued:

```
OPENIT failed to find or open [1]
- Enter new file name, including extension
```

where [1] is *Root*.`AI`. Note that entering a new filename will not change the previously defined string *Root*, and output files will still follow the naming convention of section 3.1.

If a hull component is truncated, i.e., $A_{21} \geq 0$ through calculation or input, the user is asked whether to cancel the calculation of residual drag:

```
Zero residual drag for truncated hull ... ([Y]/N)?
```

The default (affirmative) is appropriate for hulls with essentially parallel afterbodies, such as some towfish. However, for a hull that is truncated at a relatively small diameter, the correct response is negative. This would apply for a submarine hull truncated at the propulsor hub, e.g., as shown in figures 2 and 3.

3.10 Output

The output files from `DSSP20A` are:

Root.`AG`

This is the graphics output file; its format is determined by the `Plot` specification, section 3.7. More plot format options will be introduced in future releases of the code.

Root.`AL`

This is the run log output file. It contains the input records prefixed by '>> ', error messages (appendix C), and some internal run information. Its chief purpose is to aid error checking. The header and an example fragment from a typical example run log are:

```
DSSP20A log for filename root: dsub
...
Lifting component   1 output to file

>> Lift                  " Tailplane - lower rudder position
>>    Label #LowerRudder
>>    Save
>>    Tail   0.117905 0.254
>>    Planform
>>    -3.8020648 0. 0.1531236
>>    -3.8545874 0. 0.254
>>    -4.0069874 0. 0.254
>>    -4.0069874 0. 0.0826855
>>    AllMoving

?? Warning from LIFT
   (AllMoving) Default (1.0) used for S-m/S-exp
>>    ToC 0.20
>>
>> Lift                  " Tailplane - upper rudder position
Pop: invalid keyword at Level 2
.... backspace input and process component
```

The `Pop: ...` message at the end of this fragment indicates the return to Level 1 from Level 2 when an out-of-context keyword is encountered (section 3.3).

Root.AO

This is the output file to other DSSP20 modules. Like all output, it is in ASCII characters; however, as noted in section 2.1, it must not be modified by the user.

Root.AP

This is the printer output file; it is self-explanatory. Each component is described in terms of its geometric representation and derived hydrodynamic characteristics. For most purposes the output is unneccessarily large; much of it will be made optional in future releases of the code.

4 DSSP20I: INERTIAL AND ADDED MASS ESTIMATES

4.1 Input and Output Files

DSSP20I input and output files follow the nomenclature introduced in section 2.1:

File	Function
Root.AO	DSSP20A output file — required input to DSSP20I
Root.IL	DSSP20I log output file
Root.IO	DSSP20I output file to other DSSP20 modules
Root.IP	DSSP20I printer output file

There is no user-prepared input required.

4.2 Interactive Dialogue

At the start of execution, the user is asked to supply the *Root* character string which is used to identify input and output files (see section 4.1):

```
Enter file name root (default = DSSP20):
```

A string up to eight characters long is expected; it should comply with the filename requirements of the host operating system. A carriage return indicates the default "DSSP20". If the file *Root*.A0 cannot be found, the following message is issued:

```
OPENIT failed to find or open [1]
- Enter new file name, including extension
```

where [1] is *Root*.A0. Note that entering the correct name for this file will not change the character string *Root*, and output files will still follow the naming convention of section 4.1.

4.3 Output

The output files from DSSP20I are:

Root.IL

This is the run log output file. It contains error messages (appendix C) and some internal run information, principally the internal files that are used to do added mass estimation using components of the ESAM code [8]. The header and an example fragment from a typical example run log are:

```
DSSP20I log for filename root: DSSP20

...

Rewind/echo ESAM input file.

>> 1                        Lift component
>> P                        Positive X axis direction
>> 0.0                      Zero X offset
>> 0.0                      Zero Z offset
>> 0.0 1000.0               Dimensional output
>> 72                       Output columns
>> 2N  Lift 1               Neutral type 2
>>    -.511875D+01    .000000D+00 0.0 0.0
>>      .102375D+02    .000000D+00
>>      .102375D+02    .548438D+01
>>      .000000D+00    .548438D+01    .185000D+00
```

Root.IO

This is the output file to other DSSP20 modules. Like all output, it is in ASCII characters; however, as noted in section 2.1, it must not be modified by the user.

Root.IP

This is the printer output file. Results of the added mass calculations for each component are given in standard ESAM 72 column format [8]. The vehicle inertial and added mass matrices are tabulated (full scale values may overflow the edit field; this will be addressed in a subsequent release).

5 DSSP20V: VEHICLE SYSTEM DEFINITION

5.1 Input and Output Files

DSSP20V input and output files follow the nomenclature introduced in section 2.1:

File	Function
Root.VI	DSSP20V user-prepared input file
Root.AO	DSSP20A output file — required input to DSSP20V
Root.VL	DSSP20V log output file
Root.VO	DSSP20V output file to other DSSP20 modules
Root.VP	DSSP20V printer output file

5.2 Input File Specification

In *Root*.VI, Level 1 input is used to flag vehicle components and systems — descriptions of which are input at Level 2 — and to provide overall program control. Return to Level 1 from Level 2 occurs when an out-of-context keyword is encountered.

Future enhancements to DSSP20, such as autopilot control and ballast blowing, will be handled by DSSP20V and will increase the complexity of *Root*.VI. However, for the moment the file can still be relatively small, as the following example illustrates:

```
Title Example Root.VI

Prop                          !  prop type allowed to revert to default
    Diameter     4.021875
    Location   -27.7875       !  (Station 19.5)
    Right Handed
    RevLimit    295.0         !  25 kt at approx 11.7 RPM/kt
    RevRateLim   14.75        !  RevLimit/20

Lift 6                        !  use index instead of label
    DelRateLim    6.5         !  6.5 deg/s
    WeightDCI     0.  0.  0.  !  decouple sailplanes from indirect control
Lift 7
    DelRateLim    6.5
    WeightDCI     0.  0.  0.
```

Root.VI input is summarised in appendix B.

5.3 Level 1 Keywords

The following table summarizes the keyword records at Level 1 of DSSP20V input.

Keyword	Argument(s)	Function
Lift	*Label String* or NL	announces control dynamics input (Level 2)
Open		open water propeller option for WageningB type
PoD	P_P/D_P	pitch/diameter ratio for WageningB type
Prop		announces propulsor input (Level 2)
Propulsor		announces propulsor input (Level 2)
Rho	ρ	fluid density for WageningB propulsor initialization
Title	*Title String*	title for printer output; up to 64 characters
Ukt	U_s (kt)	ahead speed for (WageningB) propulsor initialization, in kt
Ums	U_s (m/s)	ahead speed for (WageningB) propulsor initialization, in m/s

These records are all optional and may appear in any order on the file, although corresponding Level 2 input must follow immediately. Prop is a synonym for Propulsor; making it a keyword allows the common abbreviation Prop to be used without ambiguity (see section 2.2).

Title *Title String*

A title string of up to 64 characters is preceeded by the keyword Title. The string is not parsed, and may therefore contain any character including delimiters and end-of-record characters. However, the normal end-of-record ("Enter" or "Carriage Return") terminates the string.

Level 2 input for the keywords Lift and Prop is discussed in sections 5.4 and 5.5 respectively; the other Level 1 keywords (Open, PoD, Rho, Ukt, and Ums) are discussed in section 5.6.

5.4 Level 2 Lifting Appendage Control Data

Level 1 keyword Lift requires a label string or index NL to identify the appendage to which the following control dynamics data apply. If the appendage is not control type AllMoving or Flapped, a fatal error is generated.

It will be seen in reference [2] that there are three methods of issuing appendage deflection commands. They are direct, in which a local deflection angle is associated with a particular appendage, quasi-direct, in which an effective deflection angle is associated with an appendage type (Sailplane, Sternplane, etc.), and indirect, in which an effective deflection angle is associated with all appendages capable of controlling the vehicle in a particular mode (depth, heading, or roll, using subcommands like Up, Starboard, etc.). The last method was provided as the mechanism for autopilot control, and appendages are assigned weights that determine the degree to which they respond in each of these indirect control modes.

The following table summarizes Level 2 appendage control dynamics input. The arguments are all real values; optional arguments are enclosed in brackets, [ ].

Keyword	Argument(s)	Function
`DelRateLim`	$\dot{\delta}_M$	control deflection rate limit
`DeltaDyn`	ζ_δ , ω_δ	control dynamic parameters
`WeightDCI`	W_{ID} , W_{IH} , $[W_{IR}]$	indirect control deflection weights

These records are:

`DelRateLim` $\dot{\delta}_M$

Argument $\dot{\delta}_M$ is the maximum control deflection rate in degree/s; it must be positive. The default value is 9 degree/s.

`DeltaDyn` ζ_δ , ω_δ

Argument ζ_δ is the nondimensional control deflection damping and ω_δ is the control response frequency in rad/s. These parameters, together with $\dot{\delta}_M$, determine control deflection time histories by way of a second order ODE for each control appendage. Damping ζ_δ must be positive, and a warning is given if it is > 1.0; the default value is 0.9. Frequency ω_δ must be positive; its default value is $10\dot{\delta}_M/\delta_M$ where δ_M is the maximum deflection angle (in v2.00, δ_M defaults to 45 degrees, but may be redefined in `DSSP20C`).

`WeightDCI` W_{ID} , W_{IH} , $[W_{IR}]$

Mandatory arguments W_{ID} and W_{IH}, and optional argument W_{IR}, are the nondimensional weights for indirect-depth, indirect-heading, and indirect-roll control respectively [2]. They multiply the commanded deflection and can be used, for example, to balance bowplane and sternplane response to depth control commands, or (with a value of zero) to decouple a particular appendage from an indirect control mode. These weights must be zero or positive, and are by default equal to 1.0.

5.5 Level 2 Propulsor Data

Level 1 keyword `Prop` or `Propulsor` announces propulsor data defined at Level 2 as summarized in the following table:

Keyword	Argument(s)	Function
`Diameter`	D_P	define propulsor diameter
`DiamRatio`	D_P/D_M	define Propulsor/hull diameter ratio
`DSSPtwo`		use `DSSP02` propulsion ratio data
`Label`	*Label String*	propulsor label
`Left`		left-handed propulsor
`Location`	x_P , $[y_P$, $z_P]$	define propulsor local origin
`Pitch`	θ_P	define local axis pitch
`RevDynam`	ζ_n , ω_n	RPM dynamic parameters
`RevLimit`	n_M	maximum RPM
`RevRateLim`	$\dot{n}_M$	RPM change rate limit

`Right`		right-handed propulsor
`STRprop`		use STR four quadrant propeller data
`WageningB`		use Wageningen B propeller data
`Yaw`	ψ_P	define local axis yaw

The `Label` definition is analogous to that of other components:

`Label` *Label String*

This record provides a user-defined propulsor label. The label string consists of a hash or question symbol ('#' or '?') followed by up to 24 alphabetic characters. DSSP20 ignores case in label strings. If there is no `Label` record, the default label is `#PROP`*nnn* where *nnn* is the number of this propulsor as encountered in the input file (i.e., *nnn* = 001 for the first, 002 for the second, etc.).

The other records are discussed below in terms of the functional group that they belong to.

5.5.1 Propulsor Type: Thrust and Torque Estimation

A record is required to specify the method of estimating the thrust and torque coefficients K_T and K_Q. In DSSP20, these coefficients are derived as functions of the advance ratio based on ship speed, $J_S = u/nD_P$ (not incorporating the wake fraction). Only one propulsor type, or method, can be selected; it must be the same for all propulsors. There are three options:

`DSSPtwo`

This is a "small main propulsor" propeller estimate approximating the model used in DSSP02 [9] for the Oberon class; i.e., an older twin-screw submarine arrangement. The model is based on simple functions of the propulsion ratio n/n_0, where n_0 is self-propulsion RPM. It may not model single screw propulsion particularly well since the propeller inflow conditions are different.

`STRprop`

This specifies a "large main propulsor" propeller estimate based on the four-quadrant single screw behind-model wind tunnel data reported in reference [10]. The propulsive efficiency $(K_T/K_Q) \cdot (J_S/2\pi)$ is unrealistically high at large J_S.

`WageningB`

This specifies a "large main propulsor" propeller estimate based on two-quadrant open water data for the Wageningen B4–70 series [11]. The method includes semiempirical calculations for wake fraction and thrust deduction, and approximates optimum pitch/diameter ratio [12].

The default method for thrust and torque estimation is `WageningB`.

5.5.2 Geometric Arrangement

A single point of application and direction for thrust and torque is required. It is generally a reasonable simplification to make these the propulsor center and shaft axis. The relevant records are:

`Location` x_P , [y_P , z_P]

The propulsor local origin, (x_P, y_P, z_P) in vehicle axes, is the point at which estimated thrust and torque are applied. y_P and z_P may be omitted, in which case they both default to zero. A `Location` record must be present.

`Pitch` θ_P

θ_P defines the pitch of the local propulsor axes with respect to vehicle axes, in degrees. The order of axis rotation is ψ_P, θ_P. If this record is omitted, θ_P defaults to zero.

`Yaw` ψ_P

ψ_P defines the yaw of the local propulsor axes with respect to vehicle axes, in degrees. The order of axis rotation is ψ_P, θ_P. If this record is omitted, ψ_P defaults to zero.

5.5.3 Handedness

The user can specify propulsor "handedness". For self-propulsion ahead, a left-handed propulsor has counterclockwise rotation looking forward, so that torque on the vehicle is positive; a right-handed propulsor has clockwise rotation, so that torque is negative. Handedness defaults to `Right`.

`Left`

The propulsor is left-handed.

`Right`

The propulsor is right-handed. This is redundant since it is the default.

5.5.4 Other Parameters

There are five other Level 2 records that can be used to define propulsor characteristics:

`Diameter` D_P

This record defines propulsor diameter D_P in metres. If it is missing, and the vehicle has only one hull component, then D_P defaults to half the maximum hull diameter; otherwise, a fatal error is generated. For one hull, both `Diameter` and `DiamRatio` records can be present so long as they are consistent within round-off error.

`DiamRatio` D_P/D_M

This record defines the ratio of propulsor diameter D_P to maximum hull diameter D_M. If it is missing, and the vehicle has only one hull component, then D_P/D_M defaults to 0.5; otherwise, a fatal error is generated. For one hull, both `DiamRatio` and `Diameter` records can be present so long as they are consistent within round-off error.

`RevDynam` ζ_n , ω_n

Argument ζ_n is the nondimensional change-of-RPM damping and ω_n is the change-of-RPM response frequency in rad/s. These parameters, together with $\dot{n}_M$, determine RPM time histories by way of a second order ODE for each propulsor. Damping ζ_n must be positive, and a warning is given if it is > 1.0; the default value is 0.9. Frequency ω_n must be positive; its default value is $10\,\dot{n}_M/n_M$ where n_M is maximum RPM.

`RevLimit` n_M

Argument n_M is maximum RPM; it must be positive. The default value is $15.433/D_P$ sec^{-1}, i.e., RPM for $J_S = 1$ at 30 kt.

`RevRateLim` $\dot{n}_M$

Argument $\dot{n}_M$ is the maximum change-of-RPM rate in RPM/s; it must be positive. The default value is 10 RPM/s.

5.6 Additional Propulsor Data at Level 1

This beta edition of DSSP20 was originally intended to allow multiple propulsor types. In amending this to allow only one type, some parameters were left at Level 1 to simplify applying them to all propulsors. This arrangement will be rationalized in future releases. The parameters in question apply only to `WageningB` propellers:

`Oper`

This record forces the open water condition of wake fraction and thrust deduction equal to zero, and may be appropriate for using `WageningB` propellers in the old twin-screw submarine arrangement. The default is to estimate wake fraction and thrust deduction for a propeller in the wake of a coaxial hull.

`PoD` P_P/D_P

This record overrides the default estimate of optimum pitch/diameter ratio for a `WageningB` propeller. It must be within the range $0.6 \geq P_P/D_P \geq 1.4$.

`Rho` ρ

This record overrides the default value of 1000 kg/m^3 for fluid density used to initialize J_S and RPM/kt for `WageningB` propellers.

`Ukt` $U_\bullet$ (kt)

This record overrides the default value of 5 kt for vehicle design speed used to initialize J_S and RPM/kt for `WageningB` propellers.

`Ums` $U_\bullet$ (m/s)

This record overrides the default value of 2.57222 m/s (= 5 kt) for vehicle design speed used to initialize J_S and RPM/kt for `WageningB` propellers.

5.7 Interactive Dialogue

At the start of execution, the user is asked to supply the *Root* character string which is used to identify input and output files (see section 5.1):

```
Enter file name root (default = DSSP20):
```

A string up to eight characters long is expected; it should comply with the filename requirements of the host operating system. A carriage return indicates the default "DSSP20". If file *Root*.AO or *Root*.VI cannot be found, the following message is issued:

```
OPENIT failed to find or open [1]
- Enter new file name, including extension
```

where [1] is *Root*.AO or *Root*.VI. Note that entering the correct name for either of these files will not change the character string *Root*, and output files will still follow the naming convention of section 5.1.

5.8 Output

The output files from DSSP20V are:

Root.VL

This is the run log output file. It contains the input records prefixed by '>> ', error messages (appendix C), and some internal run information, notably a record of propulsor J_S initialization. Its chief purpose is to aid error checking. The header and an example fragment from a typical example run log are:

```
DSSP20V log for filename root: DSSP20
...
>>   Title Example Root.VI
>>
>> Prop                          !  prop type allowed to revert to default
>>     Diameter     4.021875
>>     Location   -27.7875       !  (Station 19.5)
>>     Right Handed
>>     RevLimit   295.0          !  25 kt at approx 11.7 RPM/kt
>>     RevRateLim  14.75         !  RevLimit/20
>>
>> Lift 6                        !  use index instead of label
Pop: invalid keyword at Level 2
.... backspace input and process propulsor
...
>> Lift 7
>>     DelRateLim   6.5
>>     WeightDCI    0.  0.  0.
>>

?? Warning from SSP20V
   P/D below range (           .572712); use P/D = 0.6?
?? Execution continued by user

Initialization:
          JS            Thrust          X-force            Cost
      .950000        -2798.8614      -15463.3521      -18262.2135
     1.000000        -3872.5737      -15463.3521      -19335.9258
      .900000        -1447.1356      -15463.3521      -16910.4877
      .850000          362.8055      -15463.3521      -15100.5466
...
      .649622        15462.9113      -15463.3521          -.4408
      .649615        15463.6503      -15463.3521           .2982
      .649619        15463.2808      -15463.3521          -.0713
```

Root.V0

This is the output file to other DSSP20 modules. Like all output, it is in ASCII characters; however, as noted in section 2.1, it must not be modified by the user.

Root.VP

This is the printer output file. It contains brief summaries of input and intermediate results.

6 CONCLUDING REMARKS

This document is a user guide to the preprocessing modules of DSSP20, beta edition, with particular emphasis on input preparation. The guide is aimed at both experienced and new users. DSSP20 preprocessing modules need only be run once for any vehicle configuration, and the resulting output is a geometric and hydrodynamic vehicle description that is accessed by the tool kit library when running DSSP20-compatible client software. A user guide to currently-available client software will be published as a separate document.

The production edition of DSSP20 will contain a number of features that are incomplete in the beta edition. Validation of the semiempirical algorithms is on-going and improvements will continue to be made as new data become available. However, it is reasonable to expect that a model based on measured data, or possibly CFD analysis, will generally be more accurate for a specific vehicle. To take advantage of this, longer-term code development will explore the feasibility of providing a hybrid-calculation mode in which additional models can be readily introduced to supplement the semiempirical estimates.

APPENDIX A: Parser Dictionary List

The following 130 words are contained in the parser dictionary list, Edition 981009. Their internal representation is upper case, but user input is case-insensitive. The parser dictionary is used for input to all program modules and internally by the program.

ACONCOEFF	ACROSPIN	ADDEDMASS	ALL	ALLMOVING
BASEAREA	BG	BLOW	BOWPLANE	CAMBER
CB	CG	CLEAR	CXJ	CXZERO
DEBUG	DEL	DELRATELIM	DELTA	DELTADYN
DELTALIM	DELNONLIN	DIAMETER	DIAMRATIO	DOWN
DSSPTWO	DUMMY	ESAMVINT	FLAPGAP	FLAPPED
FLAPRODRAG	FOREPLANE	HARMONIC	HASENDP	HELP
HOERNER	HULL	HULLSEGMNT	INERTIAL	INITDELTAS
INTERPL	ISENDP	KILL	LABEL	LBOWPLANE
LEFT	LENGTH	LIFT	LIFTSEGMNT	LOCATION
LOD	LUMP	MAPLE	MIDHULL	NODEBUG
NOSE	OPEN	P	PITCH	PLANFORM
PLOT	POD	PORT	POWER	PROP
PROPULSOR	Q	R	RADIUS	REFERENCE
REFLECTED	RESET	RESIDRAGR	REVDYNAM	REVLIMIT
REVRATELIM	RHO	RIGHT	ROLL	ROLLNEG
ROLLPOS	ROTATED	ROUGHALLOW	RPM	RUDDER
SAIL	SAILPLANE	SAVE	SBOWPLANE	SHIPSTAB
SPREADSHT	STAMP	STARBOARD	START	STATIC
STATION	STBD	STERNPLANE	STOP	STRPROP
TAG	TAIL	TAILEFF	TEXT	THRUST
TIMEHISTRY	TITLE	TITLA	TITLB	TITLC
TITLI	TITLV	TOC	TOLERANCE	TORPEDO
TWIST	TYPEONE	TYPEZERO	U	UKT
UMS	UP	V	VEHICLE	W
WAGENINGB	WAKE	WEIGHTDCI	YAW	ZERO

APPENDIX B: Root.AI and Root.VI Input Summaries

The flow charts in this appendix summarise the input requirements for *Root*.AI and *Root*.VI, and illustrate how the input data streams are handled. Each flow chart comprises Level 1 input and the Level 2 input for selected Level 1 keywords (the keyword `Title` always returns to a Level 1 read) The charts, on the following pages, are:

- **B.1** *Root*.AI `Hull` input,
- **B.2** *Root*.AI `Lift` input,
- **B.3** *Root*.AI `Plot` and `Reference` input,
- **B.4** *Root*.VI `Lift` and additional propulsor input, and
- **B.5** *Root*.VI `Prop/Propulsor` input.

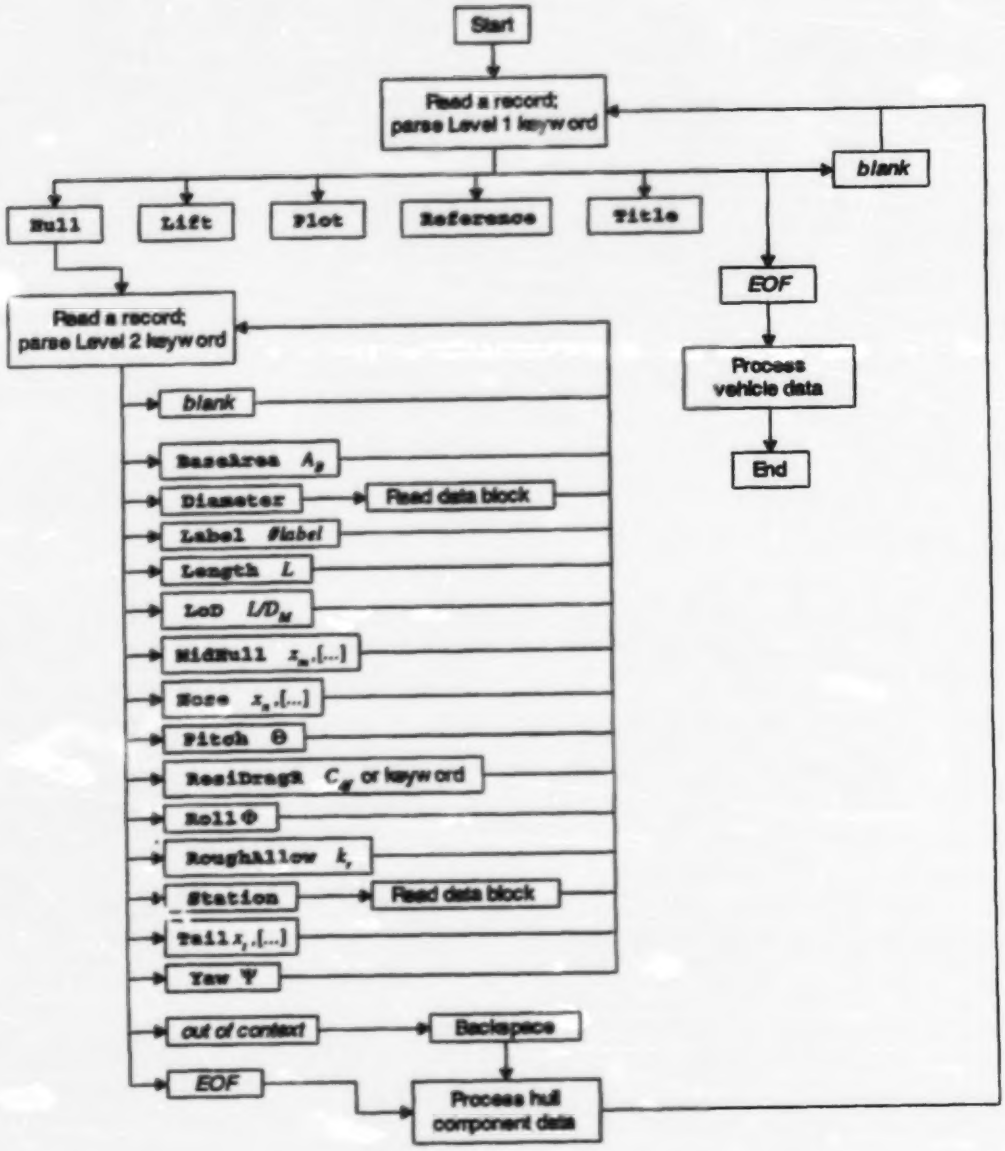

Fig B.1 *Root*.AI Hull input.

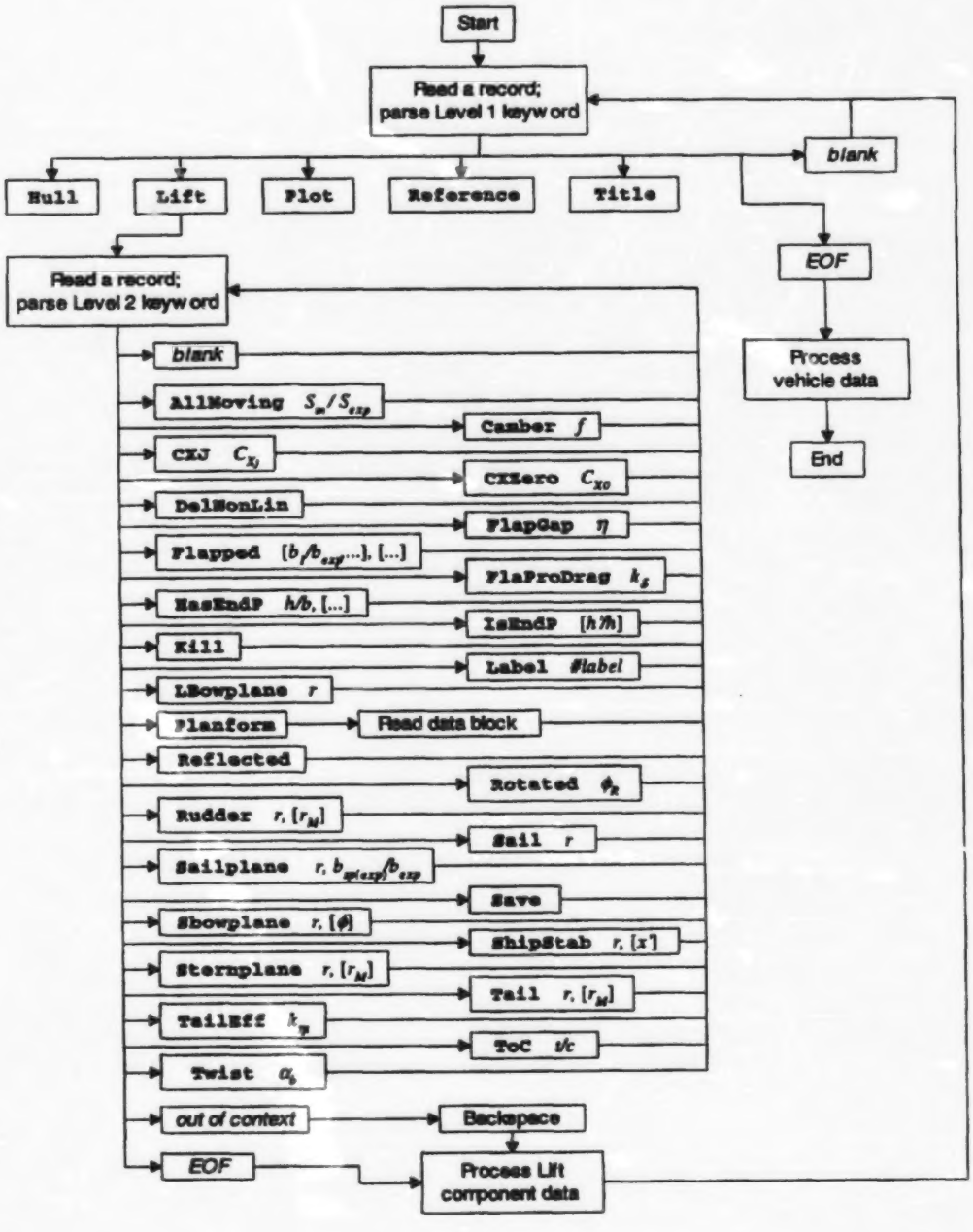

Fig B.2 *Root*.AI Lift input.

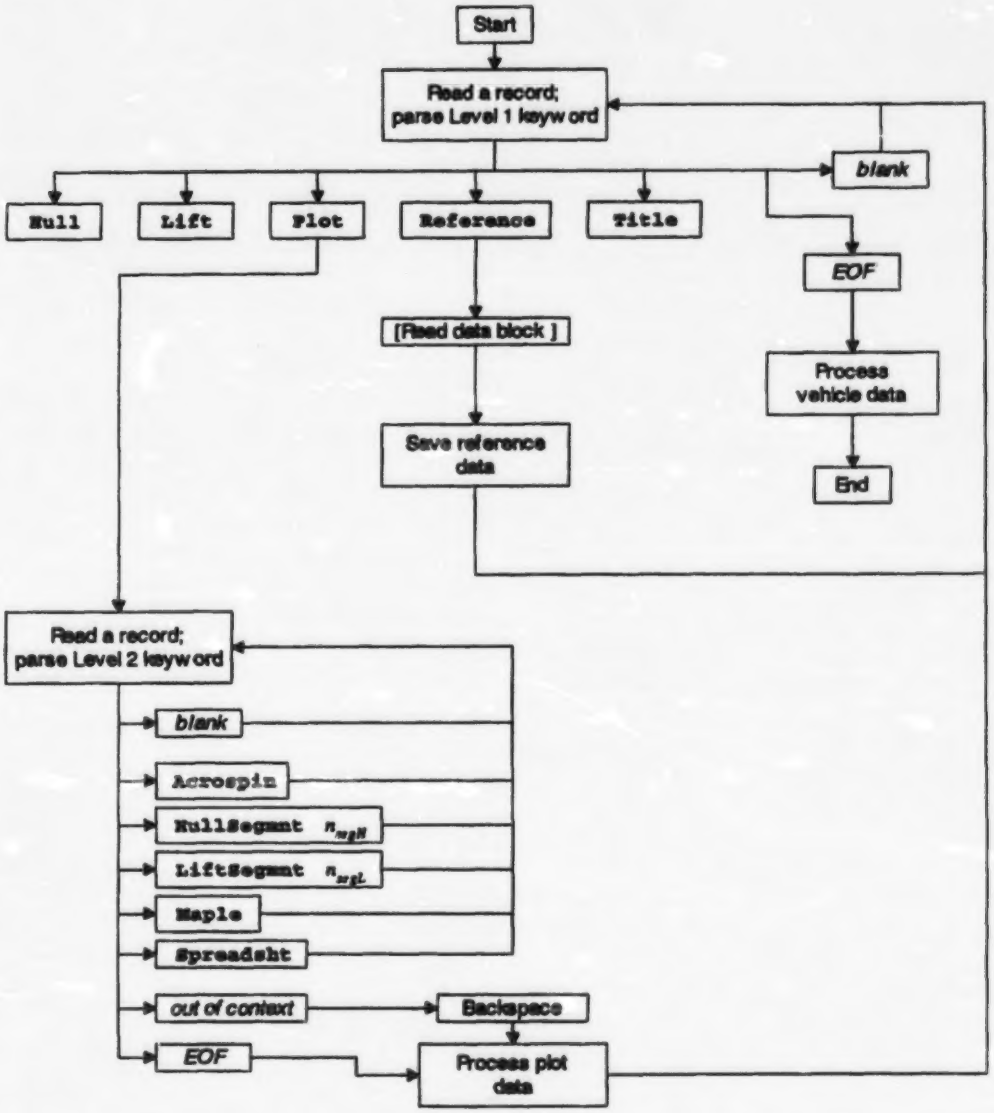

Fig B.3 *Root*.AI Plot and Reference input.

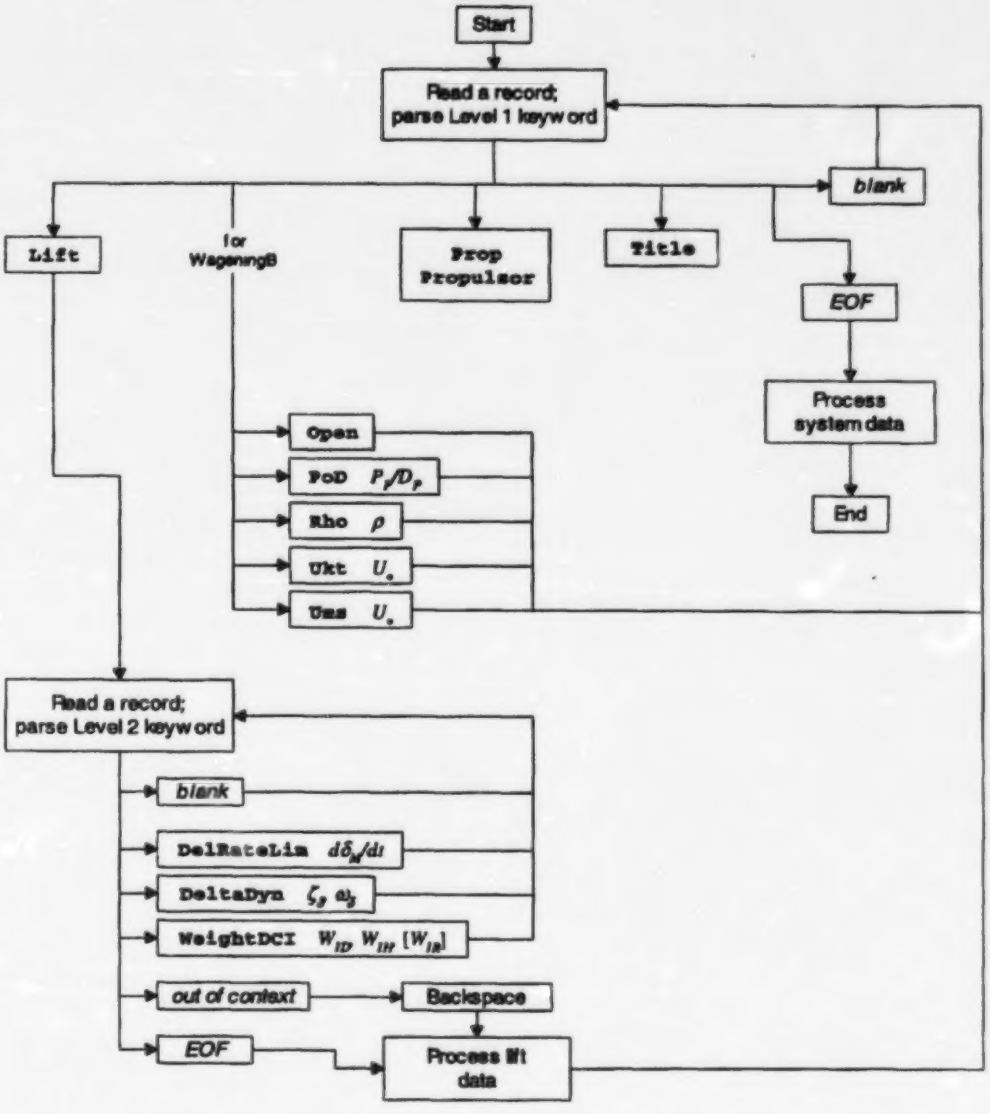

Fig B.4 *Root*.VI `Lift` and additional propulsor input.

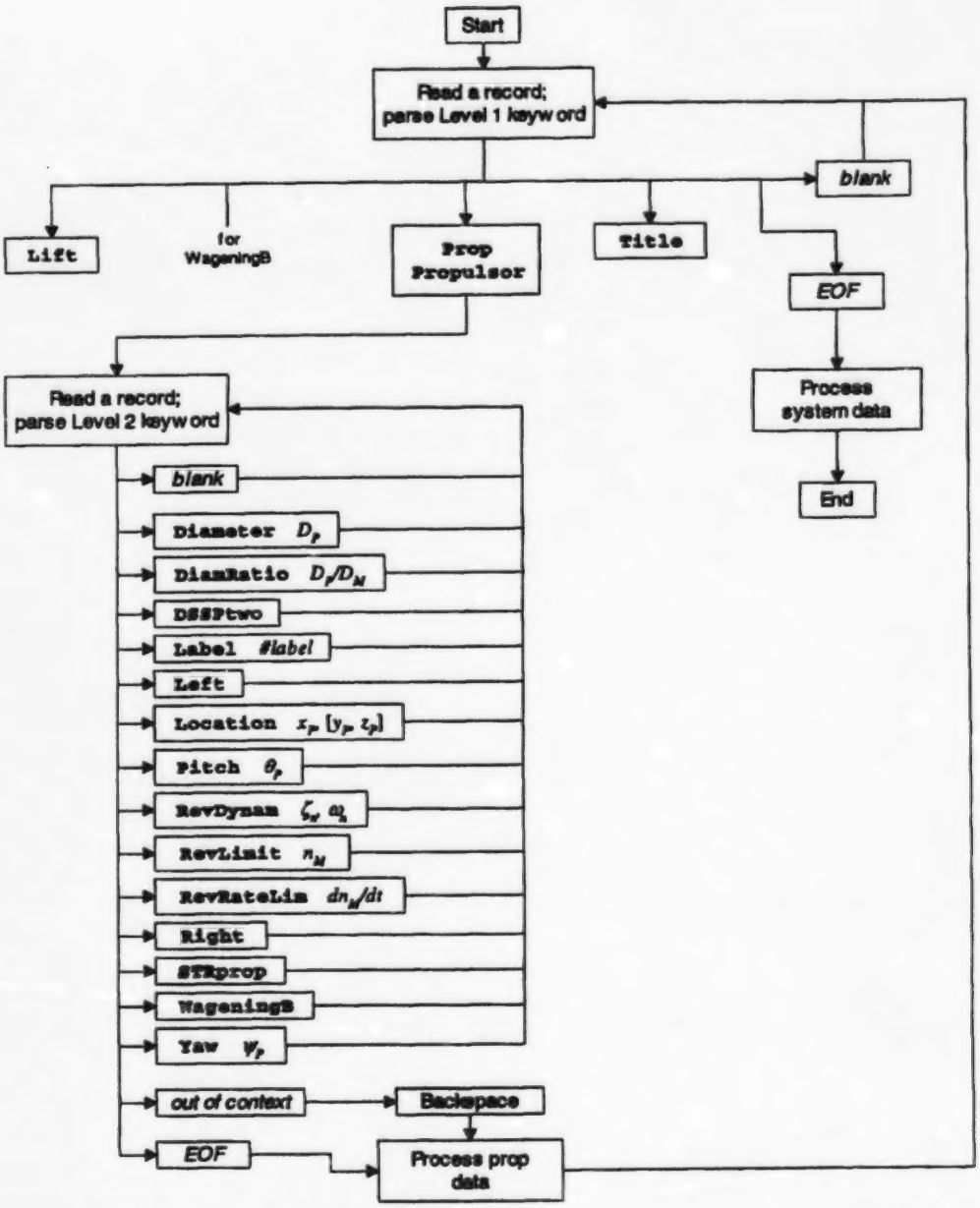

Fig B.5 *Root*.VI Prop/Propulsor input.

APPENDIX C: Error Messages

This user guide is not, except in the broadest terms, concerned with the architecture of DSSP20. However, some familiarity with it is required because error messages cite the program unit (i.e., subroutine, function, etc.) in which the error occured, thus providing a context for understanding the problem. Consequently, error messages are listed below by program unit and by the functional block, loosely corresponding to those on figure 1, in which the unit is located. These functional blocks are:

- The main program and local subprograms for DSSP20A.
- The main program and local subprograms for DSSP20I.
- The main program and local subprograms for DSSP20V.
- The core subprogram library, i.e., the tool kit.
- The utility subprogram library.

DSSP20 was developed piecemeal, using a large amount of error checking for local problems. As it was assembled, some error conditions became trapped in higher-level routines and the corresponding lower level error checks have become redundant. Errors that are believed to be redundant are indicated in the lists below with the notation "*impossible (?)*". A number of such instances may have been missed, and some may be misidentified. If a supposed redundant error is encountered, please inform the author, with the input file(s) and other supporting information, at `mackay@drea.dnd.ca`.

Some error messages are listed with the notation "*system*", indicating that they originate from compiler- or platform-related errors. These are typically 'ERR=' exits from a FORTRAN READ, WRITE, or OPEN command.

Error messages have four levels of severity: fatal, warning with pause, warning, and information There may be one or more character strings or numerical values output as part of the message. In the lists below, these are indicated by [1], [2], and so on.

Fatal error messages are generated by a condition or problem that forces termination of program execution. They are issued to both the terminal and the module log file. The format is:

```
!! Fatal Error in  program_unit_name
   message
```

A warning with pause is issued to indicate an actual or potential problem for which the user may wish to terminate program execution. These messages are issued to both the terminal and the module log file. The format is:

```
?? Warning from  program_unit_name
   message
?? Continue execution ? ... Y/[N]
```

The default response is to terminate execution. The user's response is noted in the module log file as one of:

```
?! Execution terminated by user     or
?? Execution continued by user
```

Warnings are issued to indicate an actual or potential problem while allowing program execution to continue. They are issued to both the terminal and the module log file. The format is:

```
?? Warning from  program_unit_name
   message
```

In a few instances, an information message is issued; this indicates that the program is flagging, at worst, a benign anomaly that the user may wish to consider. Information messages are issued to both the terminal and the module log file. The format is:

```
-- Information from  program_unit_name
   message
```

Many error messages are self-explanatory. This applies especially to those that arise from checking user input because input is echoed record-by-record to the module log file and the message will be displayed just after the offending input record. Typical examples from appendage parameter input to DSSP20A are:

```
!! Fatal Error in LIFT
```

followed by:

```
More than one record defining flap profile drag      or
(FlaProDrag) k-delta missing                         or
(FlaProDrag) k-delta out of range (-2.0< <0.0)
```

The error lists in this appendix are sorted in alphabetical order of, first, the program unit in which the error was flagged and, second, the error message. The code 'F', 'P', 'W', or 'I' denotes, respectively, a fatal error, a warning with pause, a warning, or an information message. The lists do not include self-explanatory messages.

C.1 DSSP20A

This functional block comprises the following program units:

SSP20A	Main program.
HULL	Process Hull Level 2 input.
HULLAX	Check hull local axes.
HULLGE	Geometric hull component calculations.
HULLHY	Hydrodynamic hull component calculations.
HULLH1	Hull linear damping calculations.
HULLOP	Hull component output to *Root*.AI.
HULLPT	Hull component output to *Root*.AG.
LIFT	Process Lift Level 2 input.
LIFTDL	Initialize control surface parameters.
LIFTGE	Geometric lift component calculations.
LIFTHY	Hydrodynamic lift component calculations.
LIFTLO	Appendage local axes calculations.

LIFTOP	Lift component output to *Root*.AI.
LIFTPT	Lift component output to *Root*.AG.
PLOT	Process Plot Level 2 input.
PLOTGE	Manage data for output to *Root*.AG.
PLOTXA	Put plotting output in AcroSpin format.
PLOTXM	Put plotting output in Maple format.
PLOTXS	Put plotting output in Spreadsht format.

The error messages from this block that may require some clarification (given in italics) are:

F SSP20A Failed to open input file
— *cannot find* Root.AI

F SSP20A Failed to open output file
— *system*

F SSP20A Failed to open plot file
— *system*

F SSP20A Failed to open plot scratch file UCHB
— *system*

F SSP20A Failed to open plot scratch file UCHC
— *system*

F SSP20A Failed to open printer file
— *system*

F SSP20A Input file error reading data
— *system*

F SSP20A Input file error reading keyword
— *system*

F SSP20A Invalid input at Level 1: [1]
— *input [1] does not match a dictionary keyword*

F SSP20A Invalid keyword at Level 1: [1]
— *keyword [1] is invalid for Level 1 input*

F SSP20A Undetermined checksum error on .AO
— *Root*.AO *failed internal self-check; impossible (?) or system*

F HULL Fewer station arguments than expected
— Diameter *or* Station *data block incomplete*

F HULL Input error reading hull parameters
— *system*

F HULL Input error reading station data
— *system*

F HULL Invalid input at Level 2: [1]
— *input [1] does not match a dictionary keyword*

F HULL (Label) - unsavable label
— Label *argument is duplicated or invalid*

F HULL (ResiDragR) invalid argument
— *unrecognized keyword argument to* ResiDragR

F HULLAX Hull input code error
— *impossible (?)*

F HULLAX Incomplete hull input
— *insufficient hull area input (see section 3.5.2)*

F HULLAX Inconsistent hull length input
— *inconsistent redundant input (see section 3.5.2)*

F HULLAX Inconsistent hull pitch input
— *inconsistent redundant input (see section 3.5.2)*

F HULLAX Inconsistent hull point input
— *inconsistent redundant input (see section 3.5.2)*

F HULLAX Inconsistent hull yaw input
— *inconsistent redundant input (see section 3.5.2)*

F HULLAX Vertical hull (... rotate)
— *hull at ±90 degrees pitch (see section 3.5.2)*

F HULLAX Zero hull length
— *colocated nose and tail*

P HULLGE Base area exceeds maximum cross section area
— *base area is outside desirable range*

W HULLGE Calculated L/D outside range (4< <16)
— *hull force and moment may be invalid for L/D outside indicated range*

F HULLGE Invalid KEYRES string
— *impossible (?)*

P HULLGE Max breadth, height, and area are inconsistent
— *these maxima are not at the same station (example: section 3.5.4)*

F HULLGE Negligible maximum cross section area
— *are areas zero in* Station *input?*

P HULLGE User-defined base area significantly different from default
— *possible discrepancy in user-specified-value*

P HULLGE User-defined L/D more than 5 percent different from calculated
— *possible discrepancy in user-specified-value*

F HULLH1 No minimum found for DT2
— *the aft hull is very bluff and has no minimum in $d(A_t)/dx$: tweak the aft hull offsets to create a minimum at the estimated location of hull separation*

F LIFT Effective flap angle for non-flapped appendage
— *inconsistent input*

F LIFT Flap gap/nose correction for non-flapped appendage
— *inconsistent input*

F LIFT Flap profile drag set for non-flapped appendage
— *inconsistent input*

F LIFT Input error reading corner points
— *system*

F LIFT Input error reading lift parameters
— *system*

F LIFT Invalid input at Level 2: [1]
— *input [1] does not match a dictionary keyword*

F LIFT (Label) - unsavable label
— Label *argument is duplicated or invalid*

P LIFT Local and maximum hull radii are equal
— *for a* Tail *(or similar) type appendage, the default tail efficiency estimation assumes that $r < r_M$*

F LIFT More than one appendage copy specification
— *only one* Reflected *or* Rotated *record is permitted*

F LIFT More than one CONTRL specification
— *only one* AllMoving *or* Flapped *record is permitted*

F LIFT More than one endplate specification
— *only one* HasEndP *or* IsEndP *record is permitted*

F LIFT More than one save configuration record
— *only one* Kill *or* Save *record is permitted*

F LIFT No planform input for a non-copied appendage
— Planform *input is required unless the appendage is copied*

F LIFT No saved configuration for copy
— *no previous appendage has been* Saved *or* Killed

F LIFT One or more parameters input for a copied appendage
— *the only valid parameter for a copied appendage is a label*

F LIFT Tail efficiency correction for non-tail appendage
— *inconsistent input*

F LIFTGE Corner point order incorrect ?
— *appendage is not oriented with leading edge ahead of trailing edge*

P LIFTGE Input twist ignored for a triangular appendage
— a Twist *record is inconsistent with zero tip chord*

P LIFTGE Input twist ignored for already twisted appendage
— a Twist *record will be ignored if* Planform *defines a twisted appendage*

F LIFTHY Max hull radius must be positive for this type
— *a positive finite* r_M *is required for Dempsey's tail efficiency estimation*

F LIFTHY (Dempsey) b/r-M < 0.402
— *Dempsey's tail efficiency estimation requires* $b/r_M < 0.402$

F LIFTLO Corner [1] is out of plane
— *impossible (?)*

F LIFTLO Internal write error
— *system write error*

F PLOT (PLOT) Invalid input: [1]
— *input [1] does not match a dictionary keyword*

F PLOT Number of hull segments must be 4,6,8,12, or 24
— HullSegmt *argument invalid*

F PLOT Number of appendage segments must be 2,4, or 8
— LiftSegmt *argument invalid*

F PLOT Input file error reading data
— *system read error at Level 2*

F PLOTGE Too many hull components
— *impossible (?)*

F PLOTGE Too many lift components
— *impossible (?)*

F PLOTGE Invalid plotting format
— *unrecognised plot type specification*

F PLOTGE Invalid or ambiguous keyword: [1]
— *impossible (?)*

F PLOTGE Error reading scratch input
— *impossible (?)*

F PLOTXS Plotting file is empty
— *impossible (?)*

C.2 DSSP20I

This function block comprises the main program SSP20I alone. For added mass calculations, it calls a subroutine ESAMCR and associated program units that were taken without modification from

the most recent version of the added mass code **ESAM** — see reference [8] — and are not discussed further here. The added mass calculations are run at arm's length and do not use standard DSSP20 error management. So far as is possible, SSP20I data set-up attempts to avoid error reporting by the ESAM components. The current version of the code runs error-free for conventional vehicle configurations.

SSP20I error messages that may require some clarification (given in italics) are:

```
F  SSP20I  Failed to open .AO input file
               — cannot find Root.AO
F  SSP20I  Failed to open ESAM input file
               — system
F  SSP20I  Failed to open output file
               — system
F  SSP20I  Failed to open printer file
               — system
F  SSP20I  (Hull) Fatal error in ESAMCR
               — STATUS = 'FATAL' returned from ESAMCR while doing a hull component
P  SSP20I  (Hull) Warning from ESAMCR
               — STATUS = 'WARN' returned from ESAMCR while doing a hull component
F  SSP20I  (Hull) Unrecognized STATUS = [i] returned from ESAMCR
               — impossible (?)
F  SSP20I  (Lift) Fatal error in ESAMCR
               — STATUS = 'FATAL' returned from ESAMCR while doing an appendage
P  SSP20I  (Lift) Warning from ESAMCR
               — STATUS = 'WARN' returned from ESAMCR while doing an appendage
F  SSP20I  (Lift) Unrecognized STATUS = [i] returned from ESAMCR
               — impossible (?)
F  SSP20I  Undetermined checksum error on .IO
               — Root.IO failed internal self-check; impossible (?) or system
```

C.3 DSSP20V

This functional block comprises the following program units:

SSP20V	Main program.
LIFTSU	Process supplementary lift control Level 2 input.
PROP	Process Propulsor Level 2 input.

The error messages from this block that may require some clarification (given in italics) are:

```
F  SSP20V  At least one propulsor expected
               — Root.VI should have at least one Prop record
F  SSP20V  Error reading .VI input
               — system
F  SSP20V  Failed to open .AO input file
               — cannot find Root.AO
F  SSP20V  Failed to open input file
               — cannot find Root.VI
```

F SSP20V Failed to open output file
— *system*

F SSP20V Failed to open printer file
— *system*

F SSP20V Incorrect or ambiguous keyword: [1]
— *input not in keyword dictionary*

F SSP20V Invalid control appendage reference
— *referenced appendage must be* AllMoving *or* Flapped

F SSP20V Invalid keyword at Level 1: [1]
— *keyword not recognized*

F SSP20V LIFT index non-integer
— *non-integer numerical argument to* Lift

F SSP20V LIFT index or label missing
— *argument required*

F SSP20V Mixed PROP parameters not allowed in v2.00
— *propulsors must have the same type and diameter*

P SSP20V P/D above range ([1]); use P/D = 1.4?
— *usually OK for* $P/D \leq 1.6$

P SSP20V P/D below range ([1]); use P/D = 0.6?
— *usually OK for* $P/D \geq 0.5$

F SSP20V Repeated control appendage data for LIFT [1] (where [1] is appendage index)
— *incorrect mix of labels and indices?*

F SSP20V Undetermined checksum error on .VO
— Root.VI *failed internal self-check; impossible (?) or system*

F SSP20V Wageningen B design speed already found
— *only one* Ukt *or* Ums *record is permitted*

F LIFTSU Error reading .VI input
— *missing data*

F LIFTSU Invalid input at Level 2: [1]
— *unmatched keyword*

P PROP Default maximum RPM (check for model scale?): [1]
— *the indicated default may be inappropriate for model scale*

F PROP (DSSPO2 or STR, one hull) D_p and D_p/D_m invalid
— D_P *and* D_P/D_M *inputs are inconsistent*

F PROP (DSSPO2 or STR, ? hulls) D_p required
— D_P *is required when there is more than one hull component*

F PROP Invalid input at Level 2: [1]
— *keyword is not in dictionary*

F PROP (Label) - unsavable label
— Label *argument is duplicated or invalid*

F PROP Location is not set
— a Location *record is required for each propulsor*

F PROP Missing propulsor data or other input error
— *a missing keyword argument or a system error*

F PROP Unrecognized TYPE: [1]
— *impossible (?)*

F PROP (Wageningen B, one hull) D_p and D_p/D_m invalid
— D_P *and* D_P/D_M *inputs are inconsistent*

F PROP (Wageningen B, ? hulls) D_p required
— D_P *is required when there is more than one hull component*

C.4 Tool Kit Library

The division of program units between the tool kit and utility libraries is somewhat arbitrary. The former contains units that are called by client software to perform the major functions of the tool kit, whereas the latter contains units that handle more general, bottom-level, functions for all parts of the program.

Both libraries consist of routines at a lower level than those in the functional blocks previously discussed. More error messages from them are listed with the notation "*impossible (?)*" because errors are generally trapped, earlier, at the higher levels of client software written by the present author. However, these errors may occur, and be correctly reported, in the case of client software from other sources.

The tool kit library comprises the following program units:

CFTURB	Turbulent flow friction drag calculation.
CFWET	Hull friction drag estimation.
DELCOM	Interpret `Del` and `Delta` control records.
DELLIM	Reset control surface deflection limits.
DYNAMI	Dynamic inertial forces for complete vehicle.
DYNAMV	Dynamic added mass forces for complete vehicle.
KTKQO2	DSSP02-based propulsion model.
KTKQST	STR four-quadrant propulsion model.
KTKQWB	Wageningen B two-quadrant propulsion model.
MAIMAT	Calculate inertial and added-mass matrices.
NREFAO	Adjust *Root*.`AO` local axes to reference axes.
NREFVO	Adjust *Root*.`VO` local axes to reference axes.
PRPCOM	Interpret `Prop` and `Propulsor` records.
PRPFRC	Propulsor force and moment calculations.
PRPINT	Initialize propulsor conditions.
PRPINX	Calculate forces for initializing propulsor.
RATCOM	Rationalize multiple commands.
READAO	Read *Root*.`AO` and set up data.
READIO	Read *Root*.`IO` and set up data.
READVO	Read *Root*.`VO` and set up data.
STATHL	Estimate hull static forces and moments.
STATLT	Estimate appendage static forces and moments.

The error messages from this block that may require some clarification (given in italics) are:

`F CFWET Error return from CFTURB; N = [1]`
— *impossible (?)*

`W CFWET Flow transition region`
— *an approximation is used for hull drag*

`W CFWET Laminar flow region`
— *an approximation is used for hull drag*

`W CFWET Re below 1000`
— *an approximation is used for hull drag*

`F CFWET Re cannot be negative`
— *impossible (?)*

F DELCOM Delta-command issued with RESET flag set
— *invalid combination of control commands*

F DELCOM Delta-command issued with ZERO flag set
— *invalid combination of control commands*

F DELCOM Invalid DELTA command argument
— *keyword argument not in dictionary*

F DELCOM LIFT([1]): Direct command flag preset
— *multiple direct commands to named appendage*

F DELCOM LIFT([1]): Indirect/Depth command flag preset
— *multiple indirect/depth commands to named appendage*

F DELCOM LIFT([1]): Indirect/Heading command flag preset
— *multiple indirect/heading commands to named appendage*

F DELCOM LIFT([1]): Indirect/Roll command flag preset
— *multiple indirect/roll commands to named appendage*

F DELCOM LIFT([1]): Quasi-direct command flag preset
— *multiple quasi-direct commands to named appendage*

F DELCOM RESET issued with command flag set
— *invalid combination of control commands*

F DELCOM ZERO issued with command flag set
— *invalid combination of control commands*

F DELLIM DELTALIM argument invalid
— *the argument must must be a label or index*

F DELLIM Invalid delta limit range
— *lower limit* $\geq$ *upper limit*

F KTKQ02 Axial speed U must be positive
— *impossible (?)*

F KTKQ02 D(iameter) must be positive
— *impossible (?)*

F KTKQST D(iameter) must be positive
— *impossible (?)*

W KTKQST (Quad 1/2) negative J out of range
— *condition:* $J_S < -1.85$

W KTKQST (Quad 1/2) positive J out of range
— *condition:* $6.15 < J_S$

W KTKQST (Quad 3/4) negative J out of range
— *condition:* $J_S < -6.15$

W KTKQST (Quad 3/4) positive J out of range
— *condition:* $2.05 < J_S$

F KTKQWB Diameter must be positive
— *impossible (?)*

F PRPCOM (Case 2a,3a) Invalid reference to PROP [1]
— *propulsor index [1] is out of range*

W PRPCOM Integer argument converted to PROP 1 RPM
— *integer argument is interpreted as RPM for a single propulsor*

F PRPCOM Internal error: lost PROP argument
— *impossible (?)*

F PRPCOM Invalid first argument
— *keyword is not in dictionary*

F PRPCOM Invalid second argument
— *keyword is not in dictionary*

F PRPFRC Propulsor index out of range
— *impossible (?)*

F PRPFRC UL is negative
— *impossible (?)*

F PRPFRC Unrecognised handedness : [i]
— *impossible (?)*

F PRPFRC Unrecognised propulsor type : [i]
— *impossible (?)*

F PRPINT Invalid flag INUAPS : [i]
— *impossible (?)*

F PRPINT No propulsors defined
— *impossible (?)*

F PRPINT Non-convergence for bracketing JS
— *tolerance for convergence may be too small*

F PRPINT PROP [i] not a main propulsor
— *propulsor axis incidence greater than 20 deg relative to reference*

F RATCOM Clear and (Reset and/or Zero) flags set
— *inconsistent control commands*

F RATCOM Command conflict on LIFT([i]) - flags D and Q set
— *inconsistent control commands*

F RATCOM Command conflict on LIFT([i]) - flags I and (D and/or Q) set
— *inconsistent control commands*

W RATCOM Deflection limited on control surface [i]
— *requested deflection exceeds limit*

F READAO Error reading .AO input
— *system*

F READAO Hull component index missing from .AO
— *impossible (?)*

F READAO Hull component index on .AO out of order
— *impossible (?)*

F READAO Incomplete TAG on .AO file
— *impossible (?)*

F READAO Invalid or ambiguous keyword: [i]
— *impossible (?)*

F READAO Invalid STAMP on .AO input file
— *Root.AO has been modified*

F READAO Lifting component index missing from .AO
— *impossible (?)*

F READAO Lifting component index on .AO out of order
— *impossible (?)*

F READAO No TAG on .AO input file
— *impossible (?)*

F READAO Number of hull components on .AO exceeds MH
— *impossible (?)*

P READAO Title-A missing ...
— *impossible (?)*

F READIO Error reading .IO input
— *system*

F READIO Hull component data miscount
— *impossible (?)*

F READIO Hull identification incorrect
— *impossible (?)*
F READIO Hull identification missing
— *impossible (?)*
F READIO Incomplete TAG on .IO file
— *impossible (?)*
F READIO Invalid STAMP on .IO input file
— *Root*.IO *has been modified*
F READIO Invalid or ambiguous keyword: [1]
— *impossible (?)*
F READIO Lift component data miscount
— *impossible (?)*
F READIO Lift identification incorrect
— *impossible (?)*
F READIO Lift identification missing
— *impossible (?)*
F READIO No TAG on .IO input file
— *impossible (?)*
F READIO TAGs on .AO and .IO files do not match
— *Root*.AO *and Root*.IO *come from different input data*
P READIO Title-I missing ...
— *impossible (?)*
F READVO AConCoeff unexpected for ML = 0
— *impossible (?)*
F READVO DelRateLim unexpected for ML = 0
— *impossible (?)*
F READVO Error reading .VO input
— *system*
F READVO Incomplete TAG on .VO file
— *impossible (?)*
F READVO Invalid STAMP on .VO input file
— *Root*.VO *has been modified*
F READVO . Invalid or ambiguous keyword: [1]
— *impossible (?)*
F READVO No TAG on .VO input file
— *impossible (?)*
F READVO Number of propulsors on .VO exceeds MP
— *impossible (?)*
F READVO Propulsor index missing from .VO
— *impossible (?)*
F READVO Propulsor index on .VO out of order
— *impossible (?)*
F READVO TAGs on .AO and .VO files do not match
— *Root*.AO *and Root*.VO *come from different input data*
P READVO Title-V missing ...
— *impossible (?)*
F READVO WageningB unexpected for MP = 0
— *impossible (?)*
F READVO WeightDcI unexpected for ML = 0
— *impossible (?)*

```
F  STATHL  COMP out of range
                — impossible (?)
F  STATHL  UL is negative
                — impossible (?)
F  STATLT  COMP out of range
                — impossible (?)
F  STATLT  Delta out of nonlinear range !
                — nonlinear flap estimate (DelNonLin) invalid for δ > 60 deg.
```

C.5 Utility Library

The utility library comprises the following program units:

ADDRNG	Add vector to double precision ring array.
ADIRNG	Add vector to integer ring array.
CHKDEC	Convert from integer binary array.
CHKINT	Convert to integer binary array.
CHKSUM	Update running checksum.
DECOMP	Matrix decomposition for solving linear system.
DEFLBL	Initialize component label array.
FCHECK	Evaluate ASCII file checksum.
GETCHR	Get character variable from input stream.
GETDBL	Get double precision value from input stream.
GETINT	Get integer value from input stream.
GTDRNG	Return vector from double precision ring array.
GTIRNG	Return vector from integer ring array.
INDICT	Locate keyword in parser dictionary.
INLABL	Locate label in component label array.
INTRPL	Interpolation of single-valued function.
MESSAG	Error message manager.
OPENIT	Open file.
QSFM	Fourth-order quadrature.
SOLVE	Linear system solver.
TOLABL	Add label to component label array.
UCCHAR	Character case conversion and filtering.
WORDIN	Find delimited character string in input stream.

The error messages from this block that may require some clarification (given in italics) are:

```
F  CHKSUM  Character translation error for:  [1]
                — impossible (?)
F  DECOMP  Singular input matrix (small time step?)
                — time step in a dynamic simulation may have become too small
F  DEFLBL  MZ exceeds I3.3 output format
                — impossible (?)
F  FCHECK  Checksum found STAMP on NEW file
                — impossible (?)
F  FCHECK  Incomplete STAMP on target file
                — target file (Root.A0, etc.) may have been modified
```

F FCHECK Invalid seed for checksum
— *impossible (?)*

P FCHECK Invalid STAMP on target file
— *target file (Root.AD, etc.) has been modified*

F FCHECK Invalid TYPE specification
— *impossible (?)*

F FCHECK STAMP missing from target file
— *target file (Root.AD, etc.) may have been modified*

F FCHECK Target file read error
— *system*

F FCHECK Undetermined checksum error
— *impossible (?)*

F GETCHR Non FORTRAN-extended character: may be a tab
— *see section 2.2*

F INDICT Duplicate dictionary entries ?
— *impossible (?)*

F INDICT Empty dictionary entry ?
— *impossible (?)*

F INDICT Incorrect abbreviation switch: [1]
— *impossible (?)*

F INLABL Duplicate component labels ?
— *impossible (?)*

F INLABL Incorrect abbreviation switch: [1]
— *impossible (?)*

F INLABL LABELS element [1] is empty
— *impossible (?)*

F INLABL LABELS element [1] is null
— *impossible (?)*

F INLABL LABELS element [1] missing label character
— *impossible (?)*

F INLABL LLABWD out of range
— *a label string must be between 2 and 25 characters long including the initial label character*

F INTRPL Data array not ascending in X
— *impossible (?)*

F INTRPL Fewer than two data points
— *impossible (?)*

F INTRPL Interpolation outside range
— *impossible (?)*

F INTRPL Zero or negative interpolation points
— *impossible (?)*

F MESSAG Incorrect MESTYP parameter .. MESSAG arguments are: [1] [2] [3]
— *(a multiline message) impossible (?)*

F QSFM Dimension of input array < 3
— *impossible (?)*

F TOLABL Filtered LABWRD is empty ?!!
— *impossible (?)*

F TOLABL INDEX out of range
— *impossible (?)*

W TOLABL Input word is a null label
— *word has only the initial label character*

W TOLABL Input word is not a label
— *initial label character missing*

W TOLABL LABWRD duplicates label [1]
— *labels must be unique*

W TOLABL LABWRD is not alphanumeric
— *label contains non-alphanumeric characters*

F TOLABL LLABWD out of range
— *a label string must be between 2 and 25 characters long including the initial label character*

F UCCHAR Incorrect filter option: [1]
— *impossible (?)*

REFERENCES

1. Mackay, M.: "Predicting the Effect of a Hull Plug on Submarine Manoeuvering using Semiempirical Methods", RINA International Symposium Warship '99, Naval Submarines 6, London, June 1999.

2. Mackay, M.: "DSSP20 (Beta Edition) — User Guide to the Simulation Modules", DREA Technical Memorandum 1999–109, August 1999.

3. Feldman, J.: "DTNSRDC Revised Standard Submarine Equations of Motion", DTNSRDC SPD–0393–09, June 1979.

4. Seto. M.L and Watt, G.D.: "The Interaction Dynamics of a Semi-Submersible Towing a Large Towfish", Eighth International Offshore and Polar Engineering Conference, Montreal, May, 1998.

5. Mackay, M.: "Estimation of the Force due to a Submarine Sail or Similar Appendage", Third Canadian Marine Hydrodynamics and Structures Conference, Halifax, August 1995.

6. Hoerner. S.F.: *Fluid-Dynamic Drag*, Published by the Author, Midland Park NJ, 1958.

7. Dempsey, E.M.: "Static Stabilty Characteristics of a Systematic Series of Stern Control Surfaces on a Body of Revolution", DTNSRDC Report 77–0085, August 1977.

8. Watt, G.D.: "Estimates for the Added Mass of a Multi-Component, Deeply Submerged Vehicle — Part I: Theory and Program Description", DREA Technical Memorandum 88/213, October 1988; Appendix B: "Program ESAM User Instructions Version 1.4" revised June 1996.

9. Mackay, M.: "Submarine Hydrodynamics Research at Defence Research Establishment Atlantic", Second Canadian Marine Dynamics Conference, Vancouver, August 1993.

10. Watt, G.D. and Fournier, E.Y.: "Submarine Propulsion Testing in the IAR 9 m Wind Tunnel", Third Canadian Marine Hydrodynamics and Structures Conference, Halifax, August 1995.

11. van Lammeren, W.P.A., van Manen, J.D., and Oosterveld, M.W.C.: "The Wageningen B–Screw Series", SNAME Transactions, Vol. 77, 1969.

12. Kennedy, J.L., DREA, Private Communication, July 1998.